Tsetlin Machine for Solving Contextual Bandit Problems

Raihan Seraj
Electrical and Computer Engineering
McGill University
Canada
raihan.seraj@mail.mcgill.ca

Jivitesh Sharma
Center for Artificial Intelligence Research (CAIR)
University of Agder
Norway
jivitesh.sharma@uia.no

Ole-Christoffer Granmo
Center for Artificial Intelligence Research (CAIR)
University of Agder
Norway
ole.granmo@uia.no

Abstract

This paper introduces an interpretable contextual bandit algorithm using Tsetlin Machines, which is an inherently interpretable machine learning method that solves complex pattern recognition tasks using propositional (Boolean) logic. The proposed bandit learning algorithm relies on straightforward bit manipulation, thus simplifying computation and interpretation. We then present a mechanism for performing Thompson sampling with Tsetlin Machine, given its non-parametric nature. Our empirical analysis shows that Tsetlin Machine as a base contextual bandit learner outperforms other popular comparable base learners on eight out of nine datasets. We further analyze the interpretability of our learner, investigating how arms are selected based on propositional expressions that model the context[1].

1 Introduction

Contextual bandits play a fundamental role in many applications involving sequential decision making, ranging from personalized recommendations [19] to designing effective treatment allocation strategies in clinical trials [11, 7]. Algorithms for contextual bandits have additionally gained significant interest because of their theoretical elegance. In brief, a decision-maker selects one of multiple bandit arms over a sequence of rounds, taking into account an observed context. Each round, the arm chosen elicits feedback in the form of a reward signal associated with the success of selecting that arm (such as a user purchasing a recommended product). The contextual bandit problem is particularly intriguing because the decision-maker must maximize the expected reward in as few rounds as possible, trading exploitation against exploration to identify the optimal arm.

In this paper, we recast the Tsetlin Machine (TM) [14] as a contextual bandit algorithm and study the resulting scheme empirically. TM is a recent machine learning approach to pattern recognition that employs a team of non-contextual bandit algorithms, in the form of Tsetlin automata [27], to learn patterns expressed in propositional logic. TMs have been shown to obtain competitive accuracy, memory footprint, and learning speed on several benchmark datasets [3, 18]. They have been particularly successful in natural language processing, including explainable aspect-based sentiment analysis [28] and robust text classification with ANDed word negations [29]. Being based on finite state automata, they further support Markov chain-based convergence analysis [30]. Leveraging the non-linear pattern recognition capability of TMs, our proposed scheme thus addresses the contextual

[1]The code is available online on: github

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

https://doi.org/10.52202/068431-2189

bandit problem using a team of non-contextual bandit algorithms.
Although existing algorithms for contextual bandits provide theoretical guarantees, they are either difficult to interpret or make assumptions that limit their usability in real-world settings. Motivated by these limitations, our work contributes as follows:

- We investigate TM as a base learner for contextual bandit problems and empirically demonstrate its effectiveness compared to other popular comparable base learners.
- We propose how the Thompson sampling scheme can be leveraged by TMs through bootstrapping [12].
- We provide interpretability analysis that shows how algorithms using TMs select arms based on propositional expressions of context features.

To the best of our knowledge, this is the first reported work on learning arm selection strategies for contextual bandits expressed in propositional logic. The paper is organized as follows. Section 2 and 3 present the problem formulation and introduces the TM algorithm, respectively. In Section 4, we show how TM can be used as a base learner for contextual bandits. The numerical results and the interpretability analysis are then presented in Section 5 and 7, respectively. We conclude the paper and provide directions for future work in Section 9. A literature overview on contextual bandits and different algorithms for solving them can be found in Appendix A.

2 Problem Formulation

We consider an online stochastic contextual bandit setup where at time t a context-reward pair denoted by (s_t, r_t) is sampled independently from past data distribution $\mathcal{D}$. Here, $s_t \in \mathcal{S}$ represents an M dimensional context vector and $r_t = ((r_t(1), \ldots, r_t(K)) \in \{0,1\}^K$ is the reward vector for K possible actions. The learner chooses an arm $u_t \in \{1, \ldots, K\}$ after observing the context s_t and receives a reward $r_t(u_t)$ for the chosen arm. The objective of the learner is to perform a sequence of actions in order to minimize the cumulative expected regret given by

$$Regret = \mathbb{E}\Big[\sum_{t=1}^{T}\Big(r_t(\pi^*(s_t)) - r_t(u_t)\Big)\Big]. \tag{1}$$

Here, $\pi^* = \underset{\pi\in\Pi}{\arg\max}\, \mathbb{E}_{(s,r)\sim\mathcal{D}}[r(\pi(s))]$, where Π denotes the set of large (possibly infinite) policies and $\pi : \mathcal{S} \mapsto \{1, \ldots, K\}$. For our analysis we consider maximizing the expected total reward which is equivalent to minimizing regret.

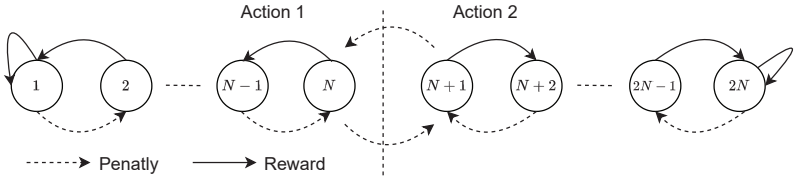


Figure 1: A two-action Tsetlin Automaton with $2N$ states.

3 Tsetlin Machine

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:

$$C_j(\mathbf{x}) = \bigwedge_{l_k \in L_j} l_k. \tag{2}$$

https://doi.org/10.52202/068431-2189

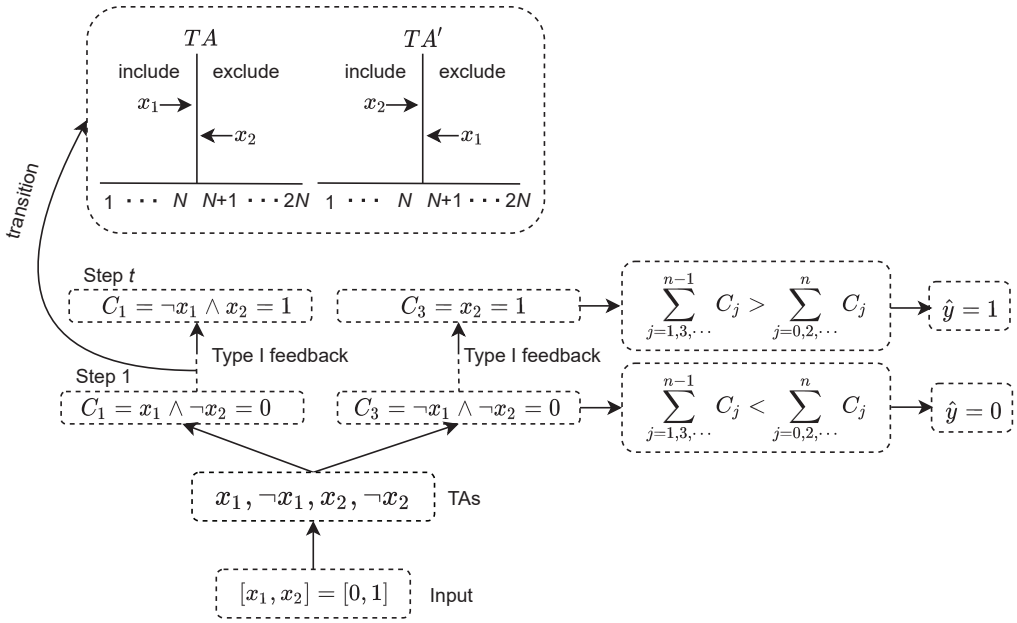


Figure 2: TM learning dynamics for an XOR-gate training sample, with input $(x_1 = 0, x_2 = 1)$ and output target $y = 1$.

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 $C_j(\mathbf{x}) = x_1 \wedge \neg x_2$ 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 Tsetlin Automata (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 get $n \times 2o$ TAs. We organize the states of these in an $n \times 2o$ 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:

$$C_j(\mathbf{x}) = \bigwedge_{l_k \in L_j} l_k = \bigwedge_{k=1}^{2o} \left[g(a_k^j) \Rightarrow l_k \right]. \tag{3}$$

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. The complexity of the algorithm is proportional (linear) to the number of features (input bits). E.g., doubling the number of features doubles the number of computations[1].

Classification. Classification is performed as a majority vote. The odd-numbered half of the clauses vote for class $\hat{y} = 1$ and the even-numbered half vote for $\hat{y} = 0$:

$$\hat{y} = 0 \leq \sum_{j=1,3,\ldots}^{n-1} \bigwedge_{k=1}^{2o} \left[g(a_k^j) \Rightarrow l_k \right] - \sum_{j=2,4,\ldots}^{n} \bigwedge_{k=1}^{2o} \left[g(a_k^j) \Rightarrow l_k \right]. \tag{4}$$

As such, the odd-numbered clauses have positive polarity, while the even-numbered ones have negative polarity. 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, $C_1(\mathbf{x}) = x_1 \wedge \neg x_2$ and $C_3(\mathbf{x}) = \neg x_1 \wedge \neg x_2$ (the negative polarity clauses are not shown). Both of the clauses evaluate to zero, leading to class prediction $\hat{y} = 1$.

Input	Clause	1		0	
	Literal	1	0	1	0
Include Literal	P(Reward)	$\frac{s-1}{s}$	NA	0	0
	P(Inaction)	$\frac{1}{s}$	NA	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P(Penalty)	0	NA	$\frac{1}{s}$	$\frac{1}{s}$
Exclude Literal	P(Reward)	0	$\frac{1}{s}$	$\frac{1}{s}$	$\frac{1}{s}$
	P(Inaction)	$\frac{1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P(Penalty)	$\frac{s-1}{s}$	0	0	0

Table 1: Type I Feedback

Input	Clause	1		0	
	Literal	1	0	1	0
Include Literal	P(Reward)	0	NA	0	0
	P(Inaction)	1.0	NA	1.0	1.0
	P(Penalty)	0	NA	0	0
Exclude Literal	P(Reward)	0	0	0	0
	P(Inaction)	1.0	0	1.0	1.0
	P(Penalty)	0	1.0	0	0

Table 2: Type II Feedback

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:

$$\epsilon = \begin{cases} T - v & y = 1 \\ T + v & y = 0. \end{cases} \tag{5}$$

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:

$$A_{t+1}^* = A_t + F^{II} + F^{Ia} - F^{Ib}. \tag{6}$$

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 $F^{Ib} \in \{0, 1\}^{n \times 2o}$ contains 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.[2] 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 $2N$:

$$A_{t+1} = \mathit{clip}\left(A_{t+1}^*, 1, 2N\right). \tag{7}$$

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

https://doi.org/10.52202/068431-2189

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

Weighted Tsetlin Machine: The learning of weights is based on increasing the weight 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 as per Equation 2. 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 [22]. 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 described 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.

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.

4 Contextual Bandits with Tsetlin Machines

We use TM as a contextual bandit learner that learns a mapping from context to actions incrementally with streaming data. Since both inputs, patterns and outputs of TM are represented as bits [14], each M dimensional context s_t is binarized with appropriate number of bits. This results in a B dimensional binarized context with $B \geq M$ which is fed to each TM learner. We outline appropriate choices of bits for binarization for different datasets in the next section. The TM learner, being non-parametric, has the advantage that it makes few or no assumption about the underlying functions to be learned; hence it is adaptive and exhibit high degree of flexibility. Another popular non-parametric learner for contextual bandits is the decision tree, which has been thoroughly studied in the literature [12, 26, 13]. The TM, in contrast, learns a linear combination of conjunctive clauses

Algorithm 1 Complete WTM learning process

Input: Training data batch (B, x, y) $\quad \triangleright\ B \geq 1$
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:
            $w_i \leftarrow w_i$ $\quad \triangleright$ [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:
            $w_i \leftarrow w_i$ $\quad \triangleright$ [No Change]
            Inaction
        end if
    end if
end for

in propositional logic by producing decision rules similar to the branches in decision trees [1], with the added advantage of being memory and energy efficient, computationally simple and not having a tendency to overfit the training data [18]. We consider two contextual bandit learning algorithms using TM: (i) TM with epsilon greedy arm selection; and (ii) TM with Thompson sampling.

Epsilon greedy TM: The exploration-exploitation trade-off is a fundamental problem in learning to make decisions under uncertainty. In a multi-armed bandit setting, the $\varepsilon-$greedy algorithm is one of the simplest ones. The learner either chooses the empirically best arm with probability $(1 - \varepsilon)$ (exploitation) or a random arm with probability ε (exploration). In the contextual bandit setting, given a set of contexts, the learner chooses to select the current empirically best arm with some high probability, maximizing immediate rewards. Otherwise, it selects a randomized arm with the hope of improving future rewards given the context. Variations of epsilon greedy algorithms have been explored in the literature, including decaying the ε parameter and eventually dropping the probability of choosing a random arm to zero. In our setting, we consider a fixed small value of ε for each TM learner associated with the arm.

TM with Thompson sampling: We now show how Thompson sampling can be achieved with TM. Our approach is similar to that presented in the Tree Bootstrap Algorithm in [12]. Since a TM learner is non-parametric, we use bootstrapping to simulate the behavior of sampling from a posterior distribution. At each time instant t, N context reward pairs are bootstrapped from the $D_{t,u}$ with replacement, where $D_{t,u}$ represents the set of observations (context reward pairs) for arm u and $N = |D_{t,u}|$. A TM learner is fitted to each of these bootstrapped datasets and at each time t, the

https://doi.org/10.52202/068431-2189

bootstrapping algorithm selects the arm u_t that has the maximum probability of success $\hat{p}$. For ease of exposition, we outline Thompson Sampling with bootstrapping for TMs in Algorithm 2.

Algorithm 2 Thompson Sampling with TM

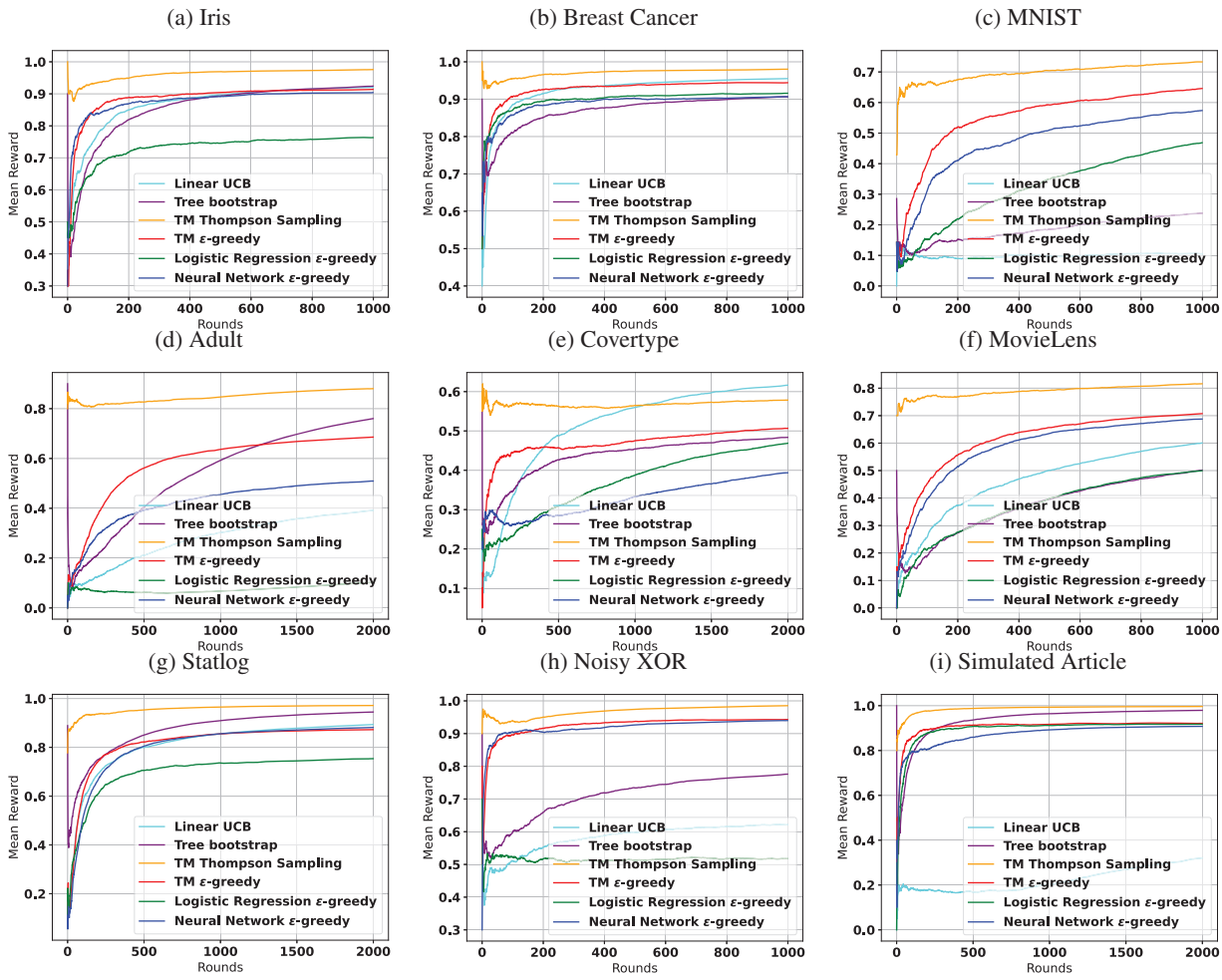
for $t = 1$ to T **do**
  get context s_t
  for $u = 1, \ldots, K$ **do**
    Sample bootstrapped dataset $\tilde{D}_{t,u}$ of size N from $D_{t,u}$ with replacement.
    Fit Tsetlin Machine $TM_{t,u}$ to $\tilde{D}_{t,u}$
  end for
  Choose action $u_t = \arg\max_u(\hat{p}(TM_{t,u}, s_t))$
  Update D_{t,u_t} with (s_t, r_{t,u_t})
end for

5 Empirical Analysis

Given the interactive aspect of contextual bandit algorithms, it is often difficult to evaluate them on real datasets except for a handful number of tasks [6]. Therefore, we use standard supervised learning classification datasets for our evaluation. We consider both binary as well as multiclass classification datasets. Each unique label in these datasets is considered as an arm of the equivalent contextual bandit problem, where a separate learner associated with each arm is trained independently. Our numerical analysis contrasts the performance of TM with $\varepsilon-$greedy arm selection and TM with Thompson sampling against Tree Bootstrap [12], Linear UCB [8], Neutral Network ($\varepsilon-$greedy) and Logistic regression ($\varepsilon-$greedy). For our analysis we consider two scenarios described as follows.
Scenario 1: In this scenario, we consider five standard supervised learning classification datasets from UCI machine learning repository [10]: *Iris, Breast Cancer Wisconsin (Diagnostic), Adult, Covertype*, and *Statlog (Shuttle)*. The response variables for each of these datasets are, *Iris types, Diagnosis, Occupation*, and *Covertypes*, respectively, where as for the *Statlog(Shuttle)* dataset, we consider the last column as the response variable. Additionally we considered two other classification datasets: *MNIST* [9] and *Noisy XOR*, where the respective response variables are *digits* and the *XOR output*. The noisy XOR dataset contains the XOR operation of 12 bit input, where the output is flipped with probability 0.4. These classification datasets are converted into a contextual bandit problem where the learner receives a reward of $+1$ for correctly identifying the target value and 0 for incorrect classification. The datasets are processed by removing entries with missing values.
Scenario 2: In this scenario, we consider two datasets: *Movielens 100 K* [15] and *Simulated Article* [23]. Both these datasets simulate a recommender system. The reward function for the Movielens dataset is the user's rating for a particular movie while for Simulated Article dataset, a reward of $+1$ is received if the recommended article is clicked by the user. The Movielens dataset is preprocessed, so that top 10 rated movies are selected. One important takeaway from these two datasets is that only the reward function for correct recommendation is provided. For instance the Simulated Article dataset provides recommended articles that has been clicked by the user. Similarly, the Movielens dataset provides user movie ratings, where the users only rated the movies they have watched. In such circumstances, the rewards are partial and sparse since there are no information on the ratings of all movies or user click information for each article. To circumvent this, we perform singular value decomposition (SVD) for both the datasets which is a popular approach in collaborative filtering [17]. For the Movielens dataset, let $S_{i,j}$ be the rating of user i on movie j. Low rank SVD of this matrix results in two matrices W and X, where $S \approx W * X$. The i^th row of W represents the context features for each user i and the rows of X represents the actions or the movie to be recommended. The reward for recommending a movie j to user i is then the dot product of the corresponding row of W_i and X_j. We perform SVD for Simulated Article dataset in a similar manner. In order to obtain binarized rewards for the actions, the maximum reward corresponding to an arm that yields a reward of 1, and the rest a reward of 0. For our analysis we consider rank 10 for both the Movielens and the Simulated Article dataset.

For both the scenarios we use appropriate binarization before fitting the TM learners. Details of binarization and parameter choices are provided in Appendix B. The results for both the scenarios are presented in Figures 3. These results show that for eight datasets, TM with Thompson sampling outperforms all other algorithms except for Linear UCB on Covertype. Further, TM with $\varepsilon-$greedy

Figure 3: Performance comparison of TM with different base learners.



provides competitive performance when compared with popular contextual bandit algorithms in the literature. Also, the TM learner learns at a faster rate compared to other learners. These experiments were performed with 10 independent runs for each dataset. The average result across the independent runs are reported.

6 Choice of Binarization Parameters

The working principle of TM relies on binarization of the context features. We use gap-based thresholding as described in [2], each threshold yielding one bit. The binarization is performed with the parameter which controls the maximum number of bits allowed to represent each feature. We varied the number of bits allowed per feature for binarization and performed an ablation study across these datasets, in order to observe the effects of sub-optimal binarization. Our analysis shows that TM as a base learner is robust to the choice of maximum bits per feature allowable for binarization. Detailed evaluation results of TM with epsilon greedy and TM with Thompson sampling are provided in Appendix C.

7 Interpretability

Unlike neural networks and some other complex machine learning approaches, one of the advantages of TMs is that they produce propositional logic expressions. These are in flat AND-form, which have proven to be human interpretable [21]. As explained in Section 3, each propositional expression is a conjunctive clause, consisting of features, in their original or negated forms, interacting with each other using logical *AND* operations. These clauses can form a simplified representation of the arm selection policy by combining them into a single Disjunctive Normal Form (DNF) expression. Since

https://doi.org/10.52202/068431-2189

clauses are assigned to each arm of the multi-armed contextual bandit problem, we produce a single DNF expression for each arm. These DNF expressions are propositional logic expressions made up of the context. The TM is able to produce these interpretations demonstrating how it interprets the context with respect to each arm.
Here, we show the simplified propositional expressions for each arm, obtained from TMs trained with Thompson Sampling on the Iris dataset:

Arm-1: $x_{10} \vee x_{14} \vee x_{15} \vee x_3$

Arm-2: $\neg x_1 \vee x_{12} \vee \neg x_{13} \vee \neg x_{14} \vee x_{16} \vee x_8 \vee \neg x_9 \vee (x_{10} \wedge x_{11} \wedge x_{15} \wedge x_2 \wedge \neg x_4 \wedge x_5 \wedge x_6 \wedge \neg x_7) \vee (x_{10} \wedge x_{11} \wedge x_{15} \wedge x_3 \wedge \neg x_4 \wedge x_5 \wedge \neg x_7)$

Arm-3: $\neg x_{10} \vee \neg x_{11} \vee \neg x_{15} \vee (x_1 \wedge \neg x_{12} \wedge x_{13} \wedge x_{14} \wedge \neg x_{16} \wedge x_2 \wedge x_3 \wedge x_5 \wedge \neg x_8 \wedge x_9) \vee (\neg x_{12} \wedge \neg x_{16} \wedge x_4)$

The next set of propositional expressions represent the arms for the Simulated Article dataset:

Arm-1: $\neg x_4 \vee (x_1 \wedge \neg x_{10} \wedge \neg x_{11} \wedge x_{12} \wedge x_{13} \wedge x_{14} \wedge x_{15} \wedge x_{16} \wedge x_{17} \wedge x_{18} \wedge x_{19} \wedge x_2 \wedge x_{20} \wedge \neg x_{21} \wedge x_3 \wedge x_5 \wedge x_6 \wedge x_7 \wedge x_8 \wedge x_9)$

Arm-2: $(x_1 \wedge \neg x_{10} \wedge \neg x_{11} \wedge x_{12} \wedge x_{13} \wedge x_{14} \wedge x_{15} \wedge x_{16} \wedge x_{17} \wedge x_{18} \wedge x_{19} \wedge x_2 \wedge x_{20} \wedge \neg x_{21} \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_6 \wedge x_7 \wedge x_8 \wedge x_9) \vee (\neg x_1 \wedge x_{10} \wedge x_{11} \wedge x_{12} \wedge x_{13} \wedge x_{14} \wedge x_{15} \wedge x_{16} \wedge x_{17} \wedge x_{18} \wedge x_{19} \wedge x_2 \wedge \neg x_{20} \wedge \neg x_{21} \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_6 \wedge \neg x_7 \wedge x_8 \wedge x_9)$

Arm-3: $(x_1 \wedge \neg x_{10} \wedge \neg x_{11} \wedge x_{12} \wedge x_{13} \wedge x_{14} \wedge x_{15} \wedge x_{16} \wedge x_{17} \wedge x_{19} \wedge x_2 \wedge x_{20} \wedge \neg x_{21} \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_6 \wedge x_7 \wedge x_9) \vee (\neg x_{11} \wedge x_{12} \wedge x_{13} \wedge x_{14} \wedge x_{16} \wedge x_{17} \wedge x_{18} \wedge x_{19} \wedge x_2 \wedge x_{20} \wedge \neg x_{21} \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_8 \wedge x_9)$

Arm-4: $x_{10} \wedge x_{11} \wedge x_{12} \wedge x_{13} \wedge x_{14} \wedge x_{15} \wedge x_{16} \wedge x_{17} \wedge x_{18} \wedge x_{19} \wedge x_2 \wedge x_3 \wedge x_4 \wedge x_5 \wedge x_6 \wedge x_8 \wedge x_9 \wedge \neg x_1 \wedge \neg x_{20} \wedge \neg x_{21} \wedge \neg x_7$

The propositional expressions shown above exhibit how the TM learns interactions between propositional input features (literals x_i). As seen, the interactions are learned with logical *AND* operations between original and negated features. These learnt interactions (clauses) are combined with the logical *OR* operation (or by addition for $T > 1$). From the propositional expressions, we can see which arm is selected by TM by simply plugging in the input values. An arm is selected if its propositional expression evaluates to True (or it obtains the largest net sum for $T > 1$). This is arguably an important advantage of TMs — a trained TM model can be reduced down to simple propositional expressions. Of course, as such, any machine learning algorithm can be converted into logical form. However, the complexity of such representations can be immense for the competitive classic machine learning models, which the TM still outperforms by a large margin [24]. Note that, although the length of expressions can increase with the increase in dimensionality of data, there are simple ways of reducing the length of expressions in $O(n)$ time [20].

8 Limitations

One of the limitations of using TM is that it requires the contexts to be binarized. While such binarization leads to a loss of information, our empirical analysis shows that TM with Thompson sampling performs substantially better than the other evaluated learners on the majority of the datasets and TM learners are quite robust to sub-optimal binarization. While our work does not provide any theoretical guarantees on performance or regret bounds for TM learners, we believe the latter will form an interesting direction of research.

9 Conclusion

In this paper, we presented an interpretable and practical contextual bandit learner using Tsetlin Machine (TM). Our analysis showed that TM as a contextual bandit learner provides competitive performance compared to other popular contextual bandit algorithms. We then presented how Thompson sampling can be implemented using TM, where our approach is derived from the Tree Bootstrap algorithm. Finally, we perform interpretability analysis where the arm selection strategy can be characterized by a propositional function of the contexts.. Having such promising empirical results, we aim at providing theoretical performance guarantees for algorithms with TM as future work.

https://doi.org/10.52202/068431-2189

References

[1] K Darshana Abeyrathna, Ole-Christoffer Granmo, and Morten Goodwin. Extending the tsetlin machine with integer-weighted clauses for increased interpretability. *IEEE Access*, 9:8233–8248, 2021.

[2] K Darshana Abeyrathna, Ole-Christoffer Granmo, Xuan Zhang, and Morten Goodwin. Adaptive continuous feature binarization for tsetlin machines applied to forecasting dengue incidences in the philippines. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2084–2092. IEEE, 2020.

[3] Kuruge Darshana Abeyrathna, Bimal Bhattarai, Morten Goodwin, Saeed Rahimi Gorji, Ole-Christoffer Granmo, Lei Jiao, Rupsa Saha, and Rohan K Yadav. Massively parallel and asynchronous tsetlin machine architecture supporting almost constant-time scaling. In *International Conference on Machine Learning*, pages 10–20. PMLR, 2021.

[4] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135. PMLR, 2013.

[5] Robin Allesiardo, Raphaël Féraud, and Djallel Bouneffouf. A neural networks committee for the contextual bandit problem. In *International Conference on Neural Information Processing*, pages 374–381. Springer, 2014.

[6] Alberto Bietti, Alekh Agarwal, and John Langford. A contextual bandit bake-off. *Journal of Machine Learning Research*, 22(133):1–49, 2021.

[7] Djallel Bouneffouf, Irina Rish, and Charu Aggarwal. Survey on applications of multi-armed and contextual bandits. In *2020 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE, 2020.

[8] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214. JMLR Workshop and Conference Proceedings, 2011.

[9] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[10] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[11] Audrey Durand, Charis Achilleos, Demetris Iacovides, Katerina Strati, Georgios D Mitsis, and Joelle Pineau. Contextual bandits for adapting treatment in a mouse model of de novo carcinogenesis. In *Machine learning for healthcare conference*, pages 67–82. PMLR, 2018.

[12] Adam N. Elmachtoub, Ryan McNellis, Sechan Oh, and Marek Petrik. A practical method for solving contextual bandit problems using decision trees. In Gal Elidan, Kristian Kersting, and Alexander T. Ihler, editors, *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, @articleTsetlin1961, author = Tsetlin, Michael Lvovitch, journal = Avtomat. i Telemekh, number = 10, pages = 1345–1354, title = On behaviour of finite automata in random medium, volume = 22, year = 1961 UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017.

[13] Raphaël Féraud, Robin Allesiardo, Tanguy Urvoy, and Fabrice Clérot. Random forest for the contextual bandit problem. In *Artificial intelligence and statistics*, pages 93–101. PMLR, 2016.

[14] 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.

[15] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[16] Insaf Ismath, KB Shashika Manosha, Samad Ali, Nandana Rajatheva, and Matti Latva-aho. Deep contextual bandits for fast initial access in mmwave based user-centric ultra-dense networks. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–7. IEEE, 2021.

https://doi.org/10.52202/068431-2189

[17] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[18] Jie Lei, Adrian Wheeldon, Rishad Shafik, Alex Yakovlev, and Ole-Christoffer Granmo. From arithmetic to logic based ai: A comparative analysis of neural networks and tsetlin machine. In *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 1–4, 2020.

[19] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[20] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[21] I. Noveck, R. B. Lea, George M. Davidson, and D. O'brien. Human reasoning is both logical and pragmatic. *Intellectica*, 11:81–109, 1991.

[22] 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.

[23] Dattaraj Rao. Contextual bandits for adapting to changing user preferences over time. *arXiv preprint arXiv:2009.10073*, 2020.

[24] Jivitesh Sharma, Rohan Kumar Yadav, Ole-Christoffer Granmo, and Lei Jiao. Drop clause: Enhancing performance, interpretability and robustness of the tsetlin machine. *CoRR*, abs/2105.14506, 2022.

[25] Yilin Shen, Yue Deng, Avik Ray, and Hongxia Jin. Interactive recommendation via deep neural memory augmented contextual bandits. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 122–130, 2018.

[26] Dennis Soemers, Tim Brys, Kurt Driessens, Mark Winands, and Ann Nowé. Adapting to concept drift in credit card transaction data streams using contextual bandits and decision trees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[27] Michael Lvovitch Tsetlin. On behaviour of finite automata in random medium. *Avtomat. i Telemekh*, 22(10):1345–1354, 1961.

[28] Rohan Kumar Yadav, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin. Human-level interpretable learning for aspect-based sentiment analysis. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21). AAAI*, 2021.

[29] Rohan Kumar Yadav, Lei Jiao, Ole Christoffer Granmo, and Morten Goodwin. Robust Interpretable Text Classification against Spurious Correlations Using AND-rules with Negation. In *The 31st International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.

[30] Xuan Zhang, Lei Jiao, Ole-Christoffer Granmo, and Morten Goodwin. On the Convergence of Tsetlin Machines for the IDENTITY- and NOT Operators. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[31] Dongruo Zhou, Lihong Li, and Quanquan Gu. Neural contextual bandits with ucb-based exploration. In *International Conference on Machine Learning*, pages 11492–11502. PMLR, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 8
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See footnote on first page.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

https://doi.org/10.52202/068431-2189