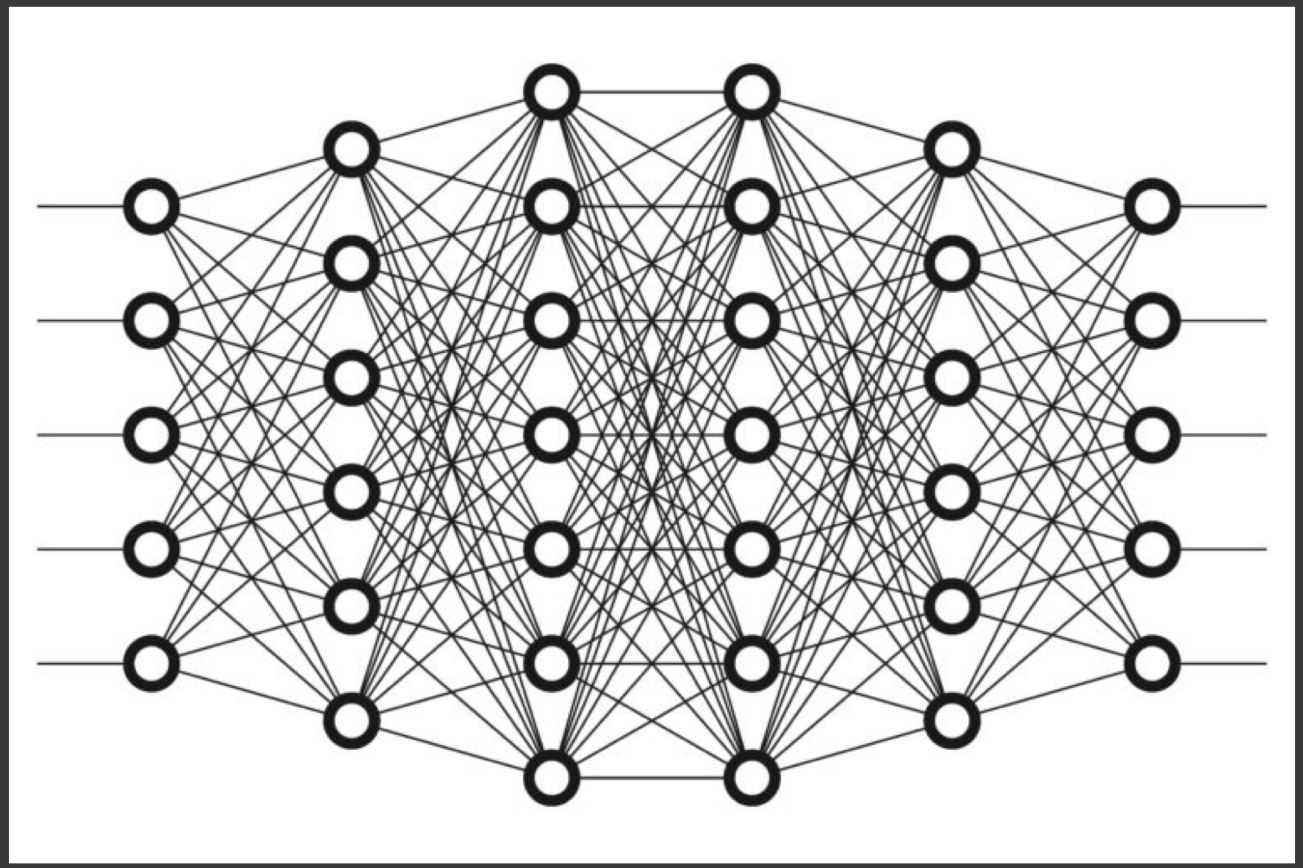
Tsetlin Machine Tutorial

https://github.com/cair/TsetlinMachine

Ole-Christoffer Granmo

September 9, 2019

The complexity of neural networks



0110110011110000111

Why Tsetlin Machines?

Interpretable

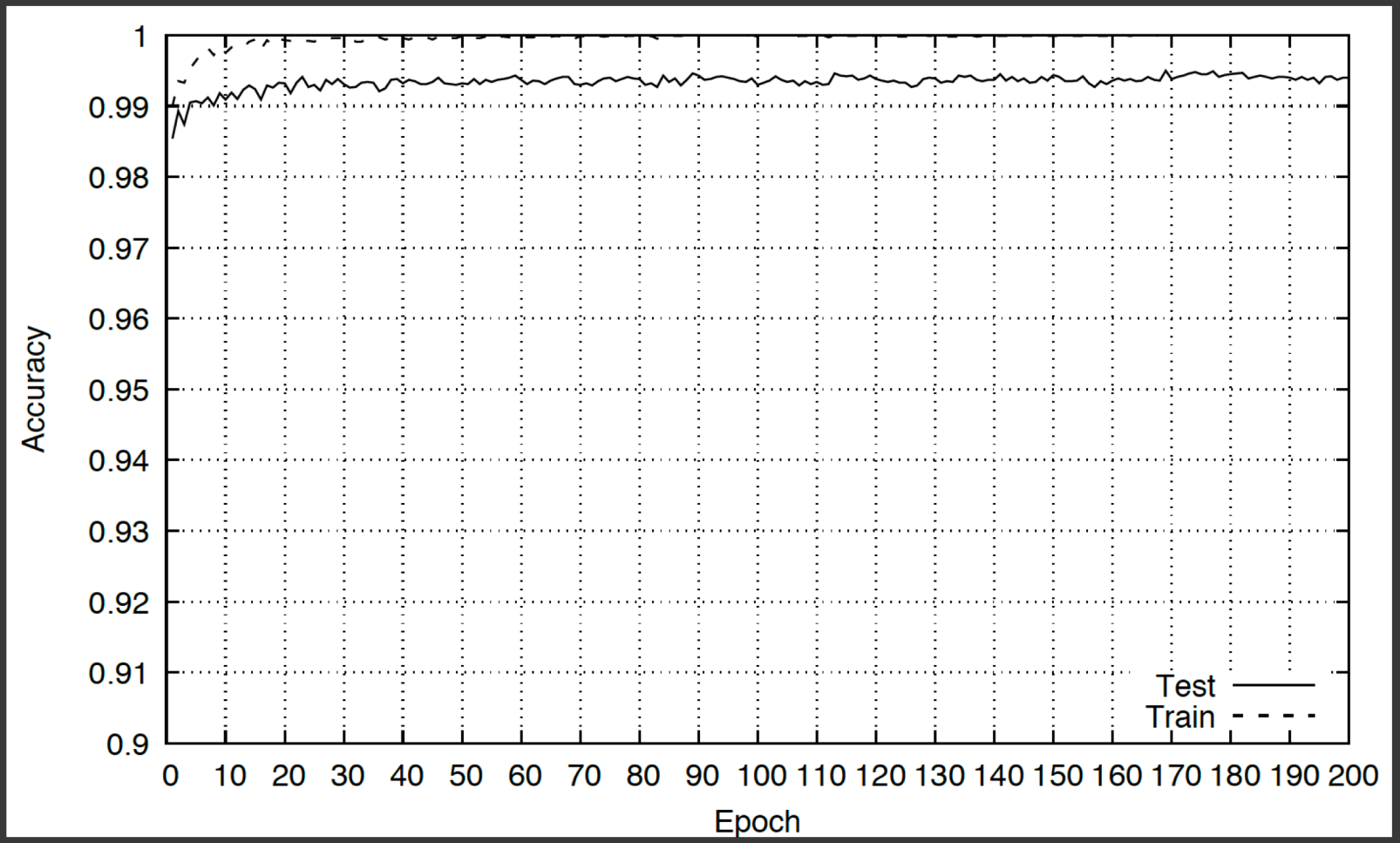
Propositional logic is the branch of logic that studies ways of joining and/or modifying entire propositions, statements or sentences to form more complicated propositions, statements or sentences, as well as the logical relationships and properties that are derived from these methods of combining or altering statements.

The Internet Encyclopedia of Philosophy

Interpretable

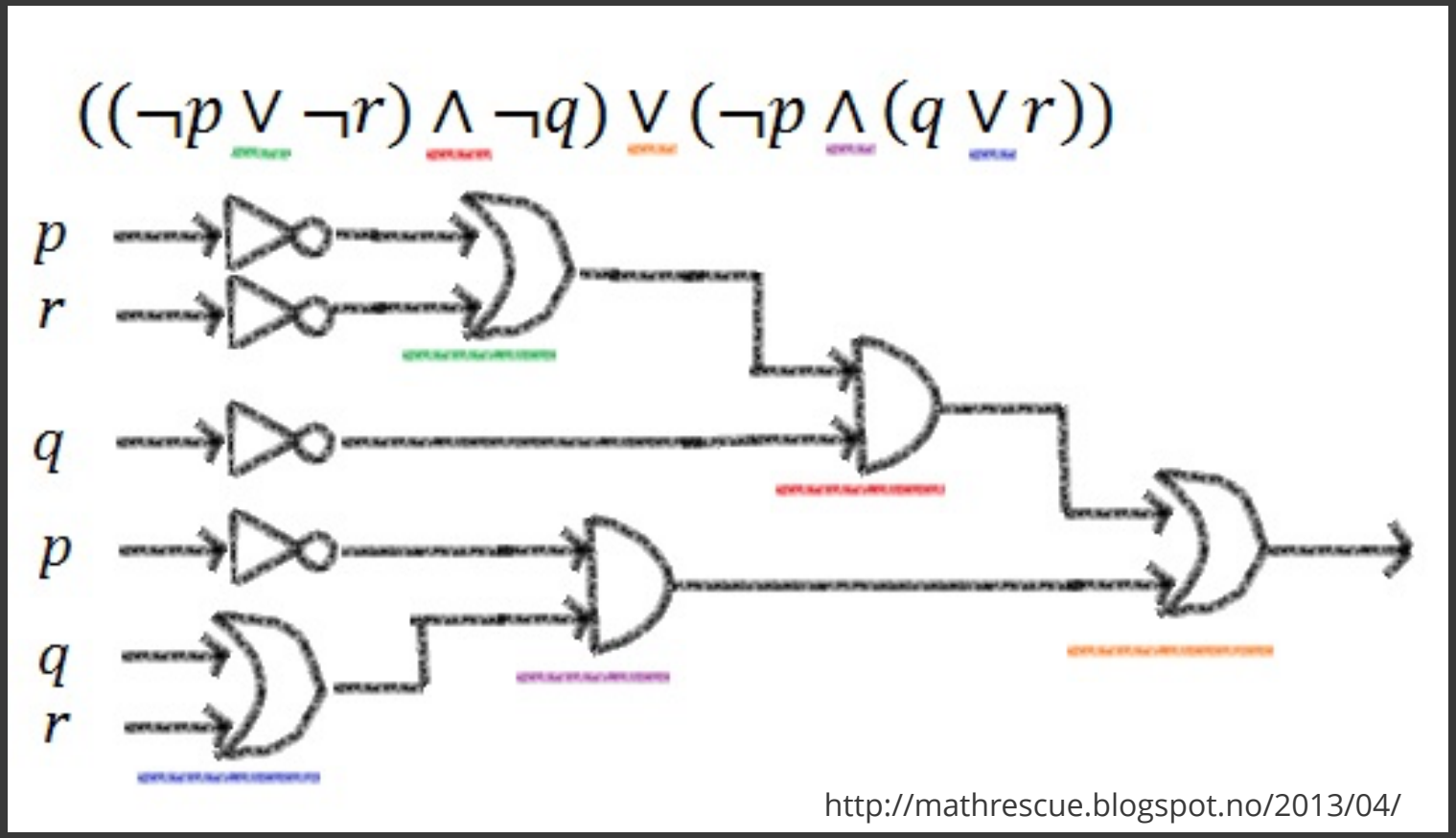
IF “grass” **AND** “asthmatic” **AND** “short of breath” **AND** “itchy skin” **AND** “symptom picture” **AND** “Cetirizin” **AND** “hyperventilating” **AND** “causes” **AND** “episodes” **THEN** Allergy

Competitive accuracy



Training and test accuracy for the Convolutional Tsetlin Machine on MNIST, epoch-by-epoch, in a single run

Hardware-near



http://mathrescue.blogspot.no/2013/04/

Newcastle University

NOKIA Bell Labs

CAIR CENTRE FOR ARTIFICIAL INTELLIGENCE RESEARCH

Propositional logic

What is a proposition?

A Proposition is an atomic sentence that can either be **True** or **False** and nothing else.

Examples:

The proposition “India is a country“ is **True**

The proposition “100 is greater than 200” is **False**

Simple and compound propositions

- A **simple proposition** is one that does not contain any other propositions as its part.
- A **compound proposition** is one that is made up of two or more simple propositions.
- We use lower case letters $x_1, x_2, x_3, \ldots$ to represent simple propositions.

Examples of simple and compound propositions

Simple: “It is raining”

Compound: “Today is Sunday **and** Sunday is a holiday”

Logical operators

Logical operators joins simple propositions into compound propositions and compound propositions into larger compound propositions.

Types of logical operators

- Negation (also called NOT)
- Disjunction (also called OR)
- Conjunction (also called AND)
- ...

Negation (NOT)

x_1	$\neg x_1$
0	**1**
1	**0**

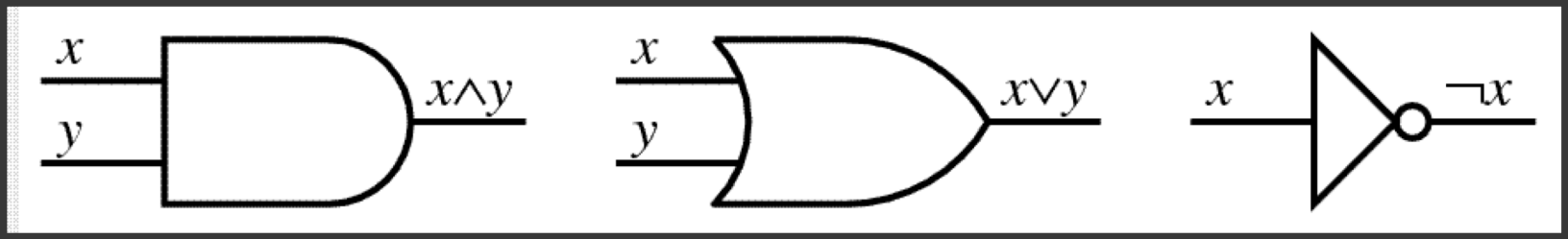
Disjunction (OR)

x_1	x_2	$x_1 \vee x_2$
0	0	0
1	0	1
0	1	1
1	1	1

Conjunction (AND)

x_1	x_2	$x_1 \wedge x_2$
0	0	0
1	0	0
0	1	0
1	1	1

Logic gates



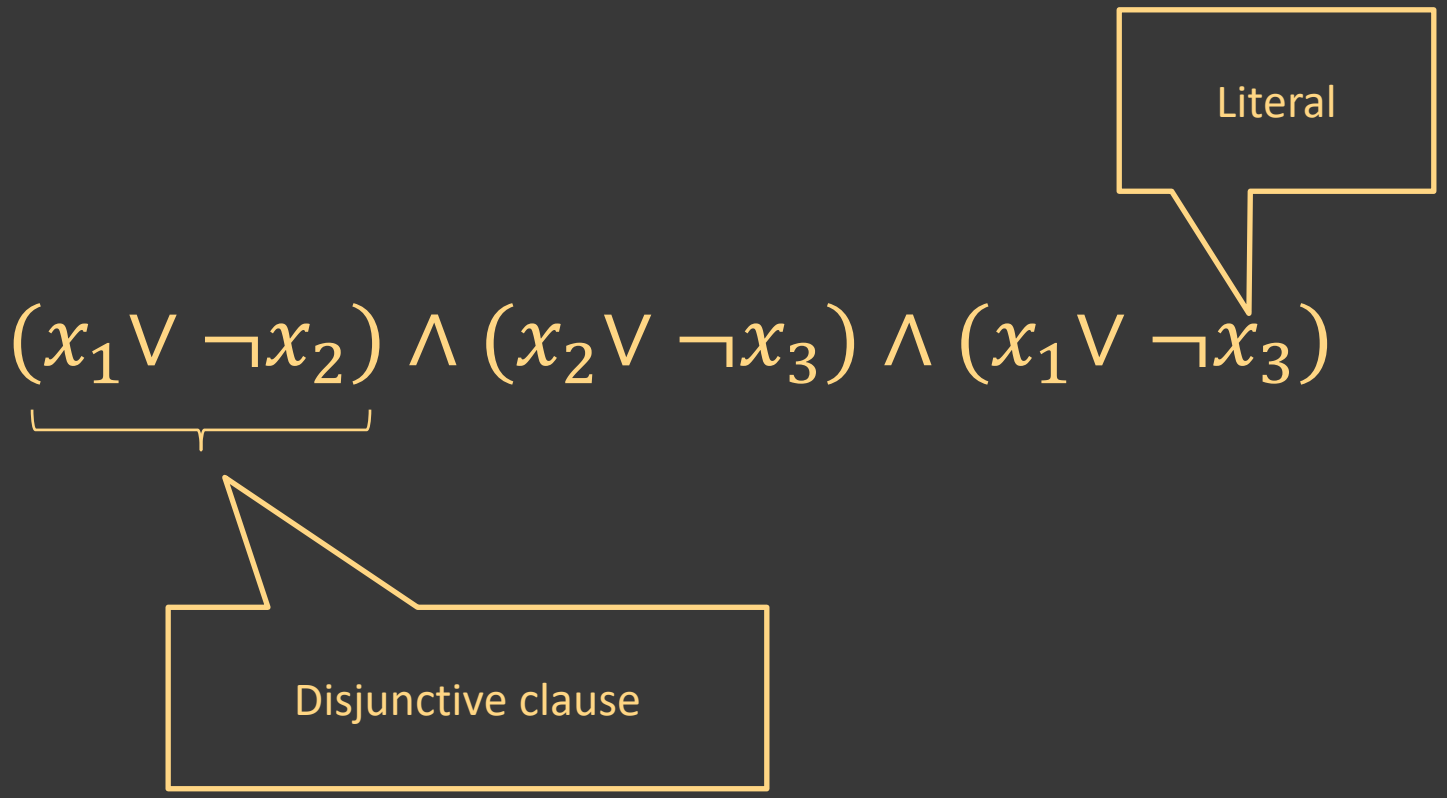
De Morgan's law

$$\neg x_1 \wedge \neg x_2 = \neg(x_1 \vee x_2)$$

$$\neg x_1 \vee \neg x_2 = \neg(x_1 \wedge x_2)$$

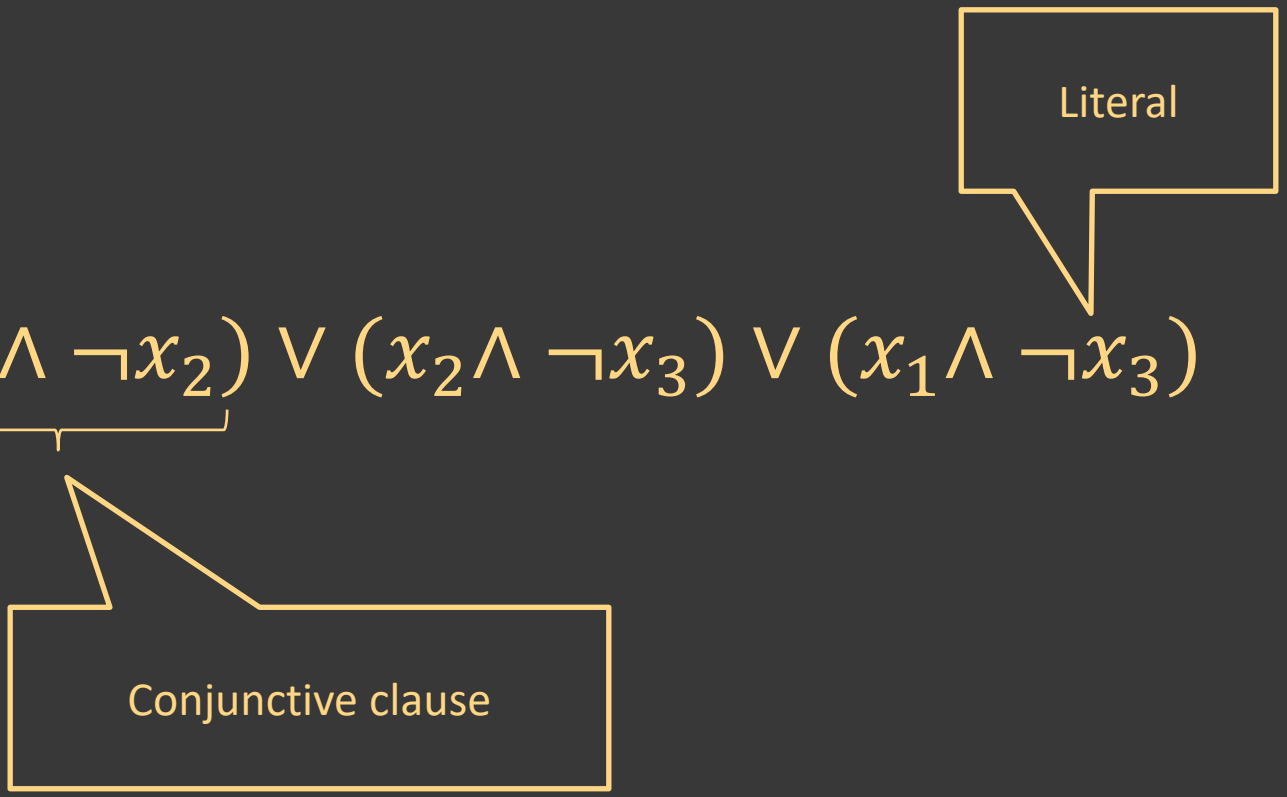
Conjunctive normal form (CNF)

Any propositional formula can be written as conjunctions of disjunctions:



Disjunctive normal form (DNF)

Any propositional formula can be written as disjunctions of conjunctions:



Exercise

What is the truth value of

$$(x_1 \wedge \neg x_2) \vee (x_2 \wedge \neg x_3) \vee (x_1 \wedge \neg x_3)$$

when

$$x_1 = 1, x_2 = 1, x_3 = 1$$

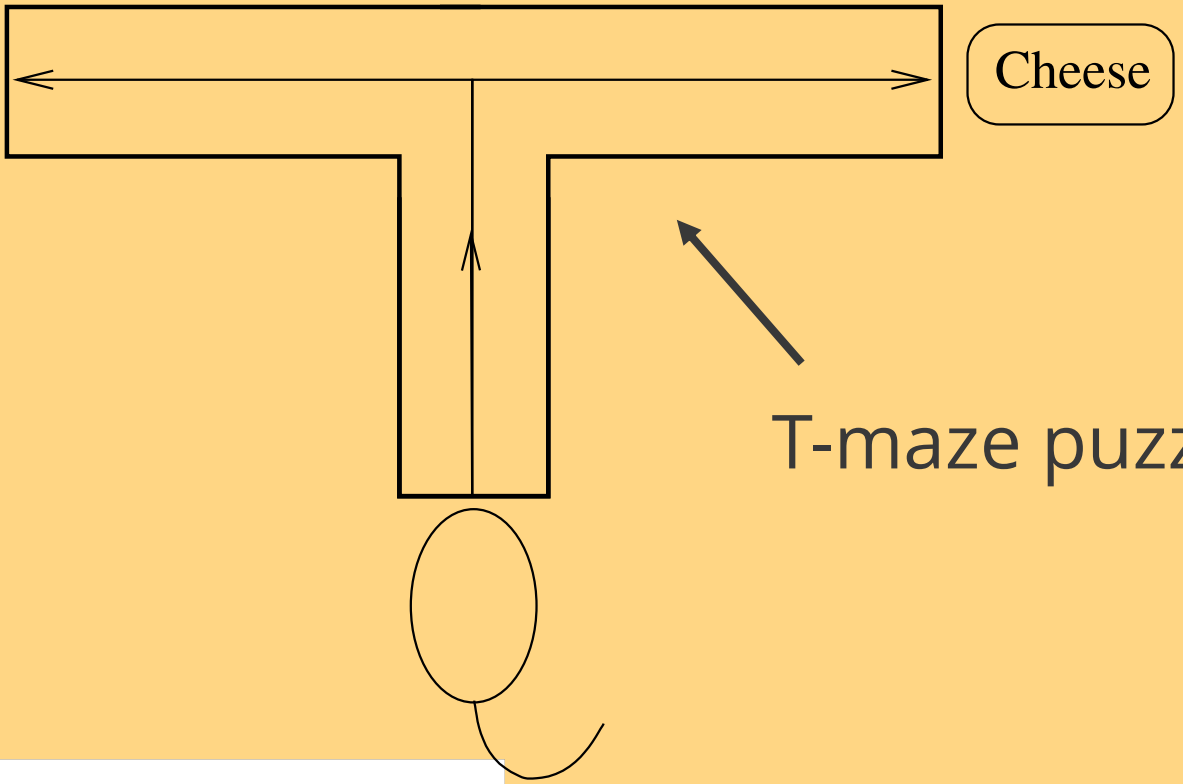
The Tsetlin automaton

Michael Lvovitch Tsetlin

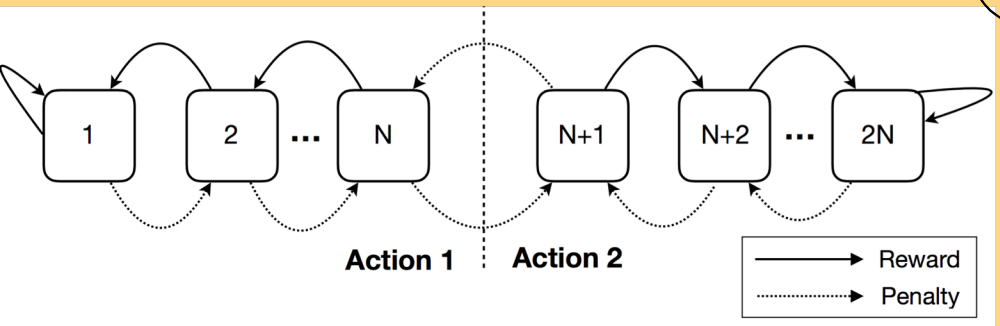
Michael Lvovitch Tsetlin (the surname is also written Cetlin, Tzetlin, Zeitlin, Zetlin; cyrillic: Михаил Львович Цетлин) (22 September 1924 – 30 May 1966) was a Russian mathematician and physicist who worked on cybernetics.

https://en.wikipedia.org/wiki/Michael_Lvovitch_Tsetlin

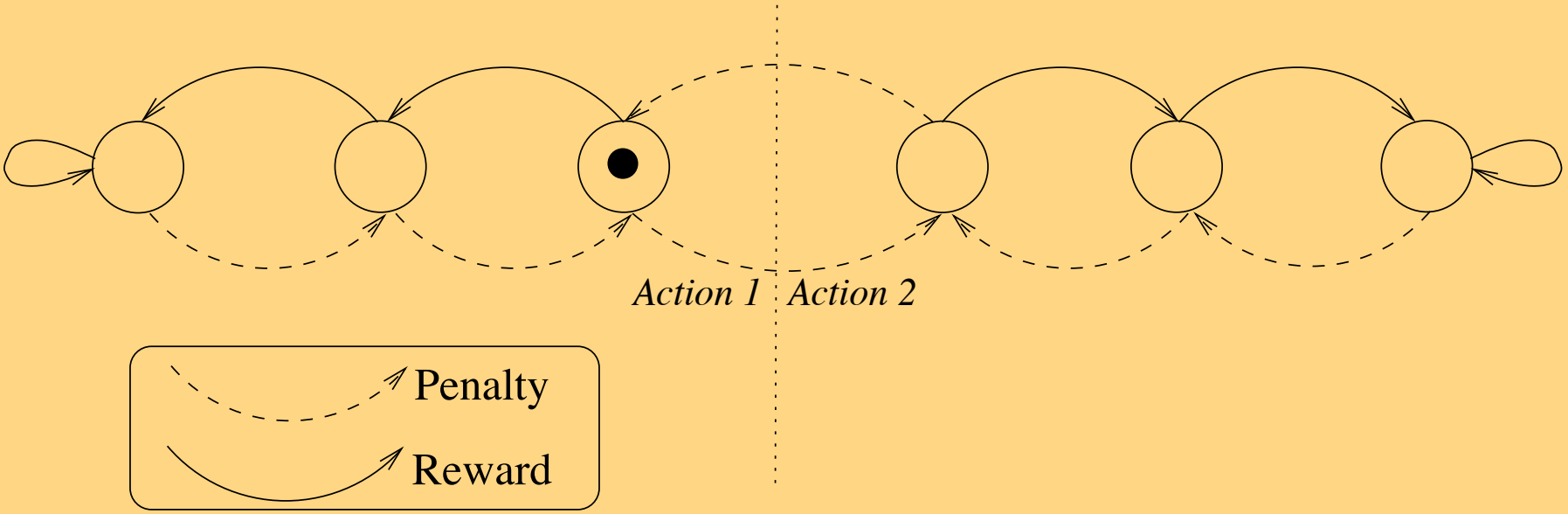
The Tsetlin automaton



T-maze puzzle

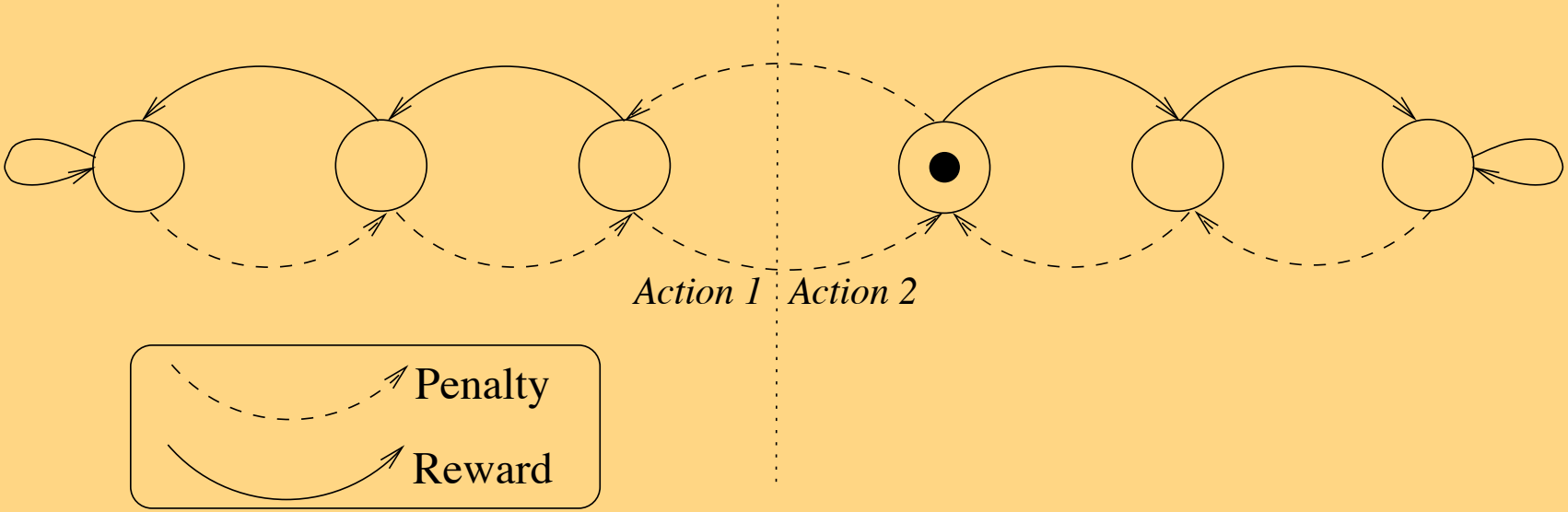


The Tsetlin automaton



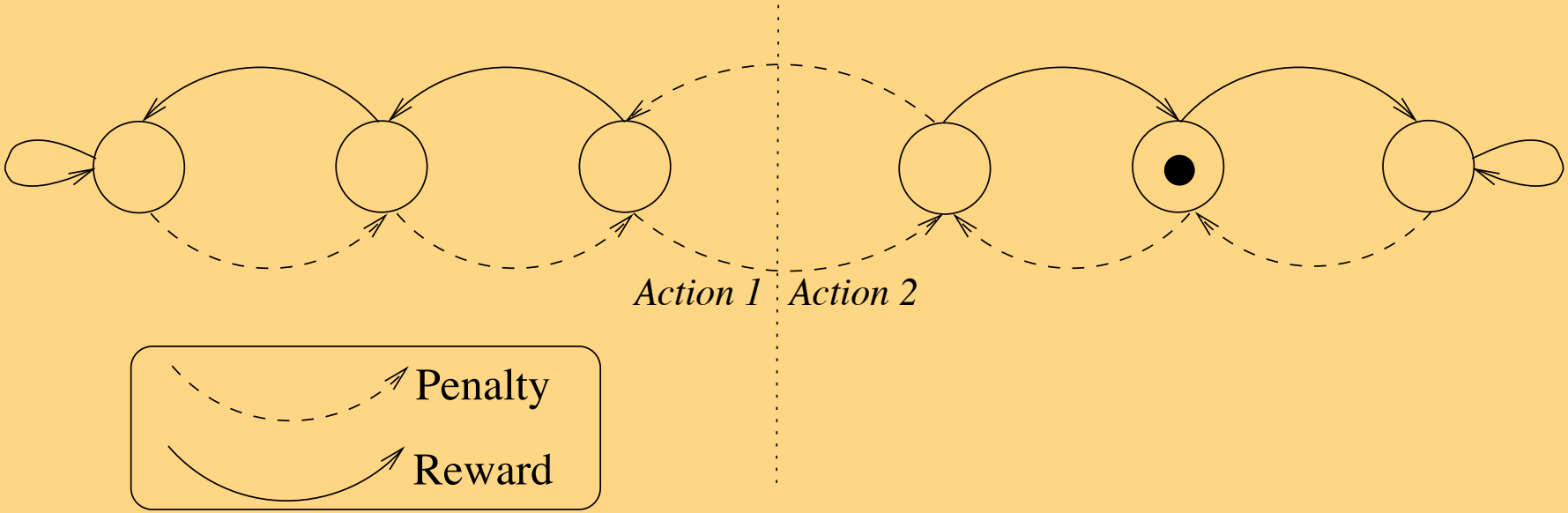
Selected Action: Action 1
Response from Environment: Penalty

The Tsetlin automaton



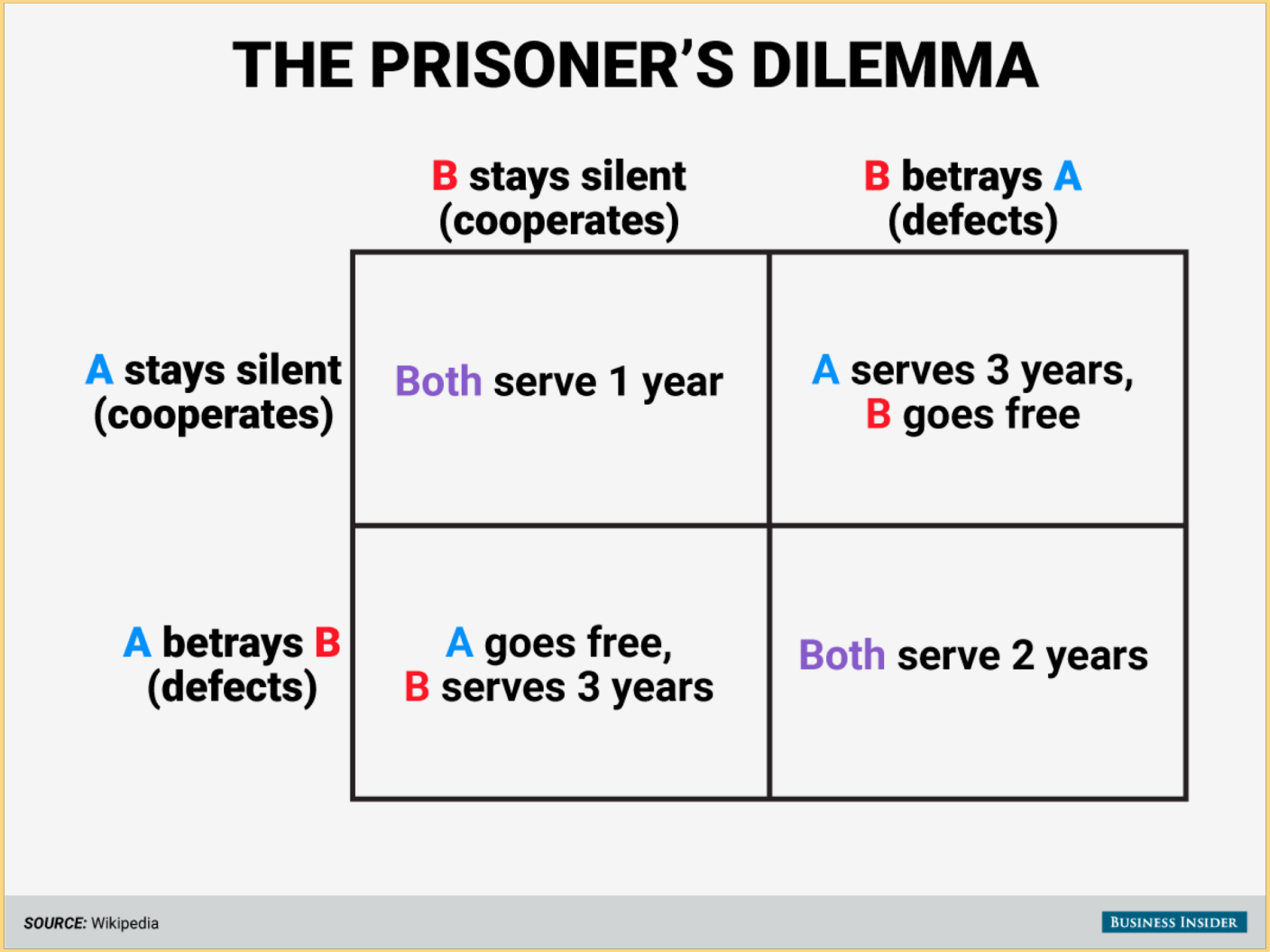
Selected Action: Action 2
Response from Environment: Reward

The Tsetlin automaton



Selected Action: Action 2
Response from Environment: Reward

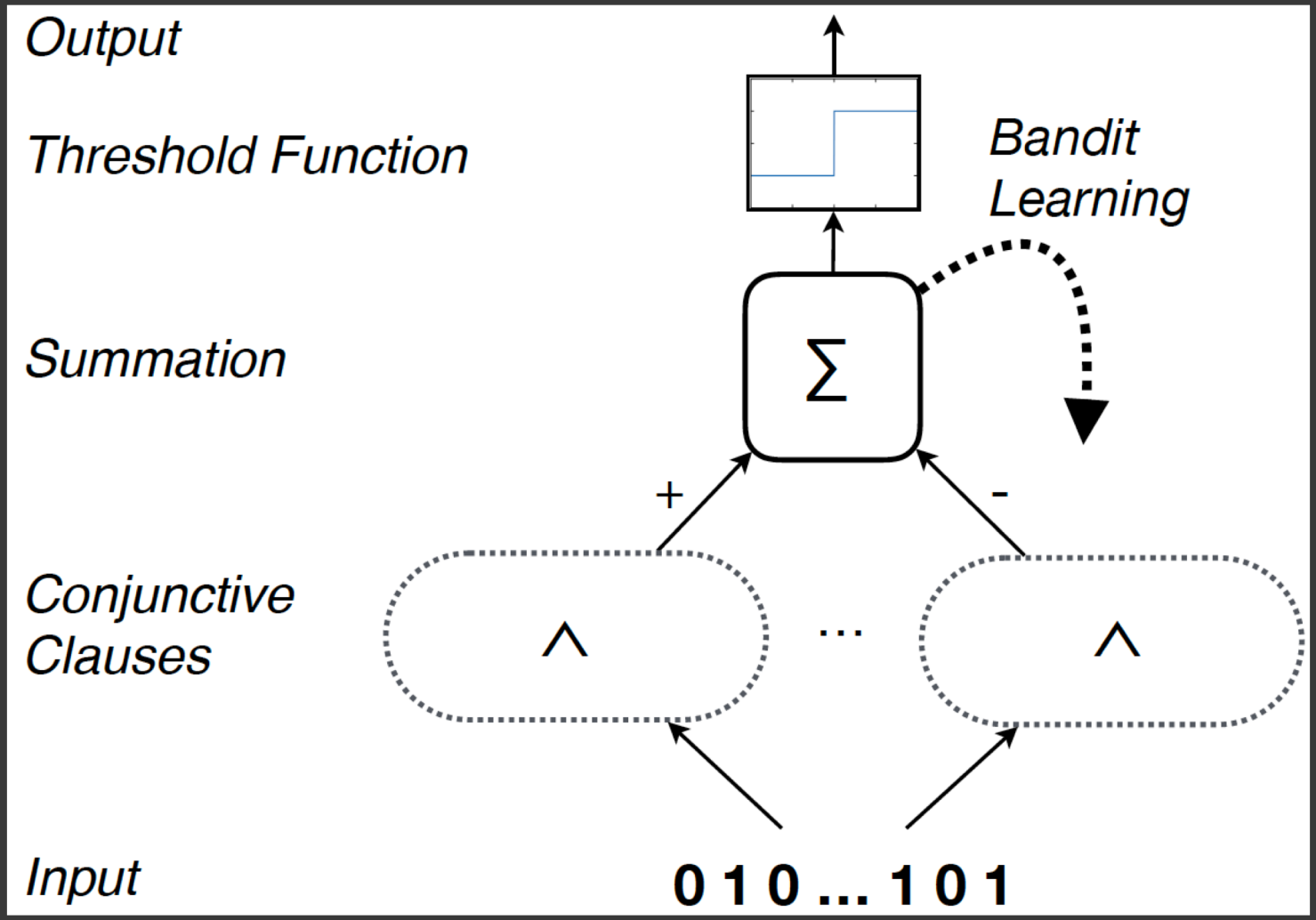
Tsetlin automata and games



The Tsetlin Machine

CAIR
CENTRE FOR ARTIFICIAL INTELLIGENCE RESEARCH

The Tsetlin Machine

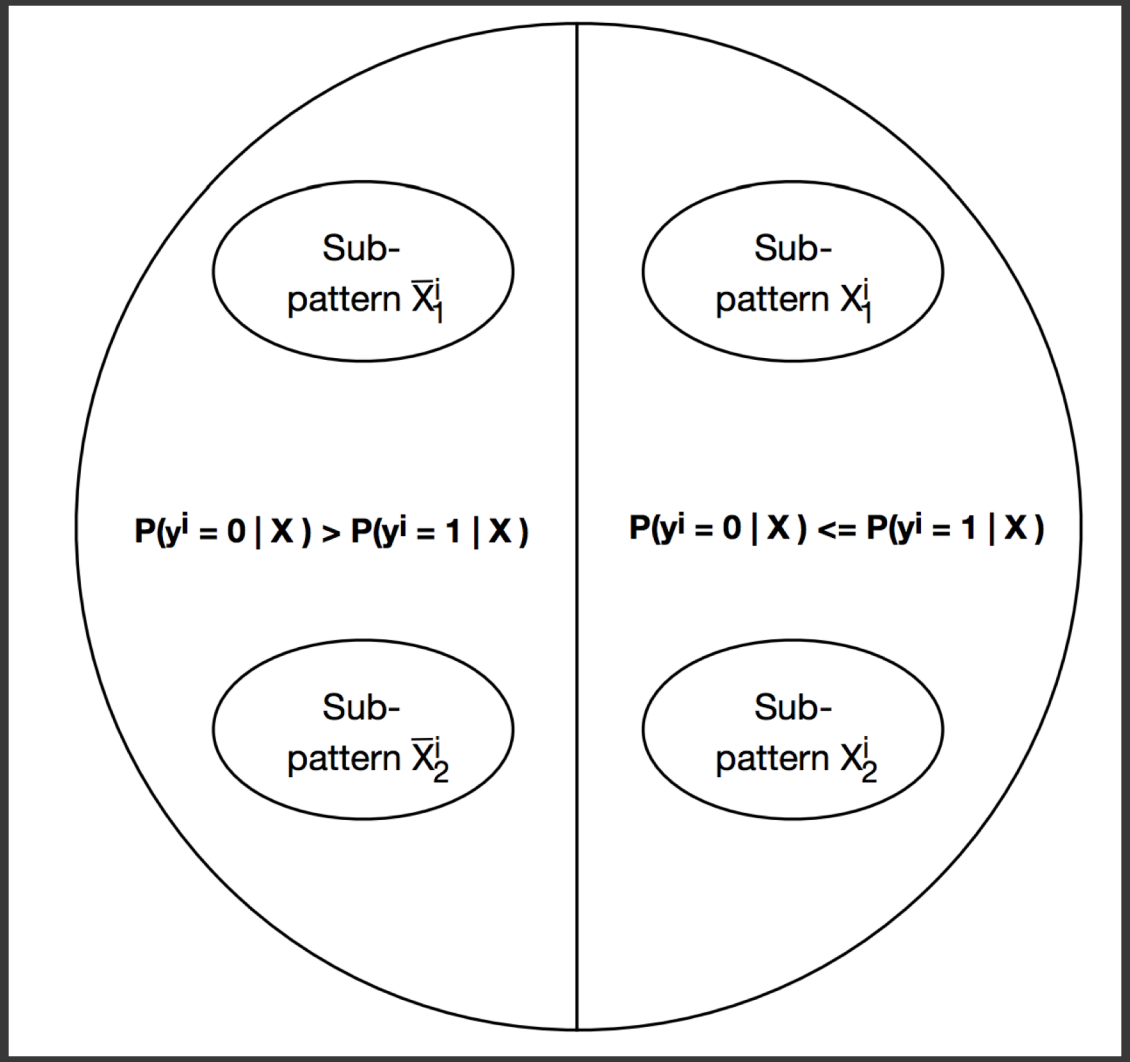


https://github.com/cair/TsetlinMachine

The pattern recognition problem

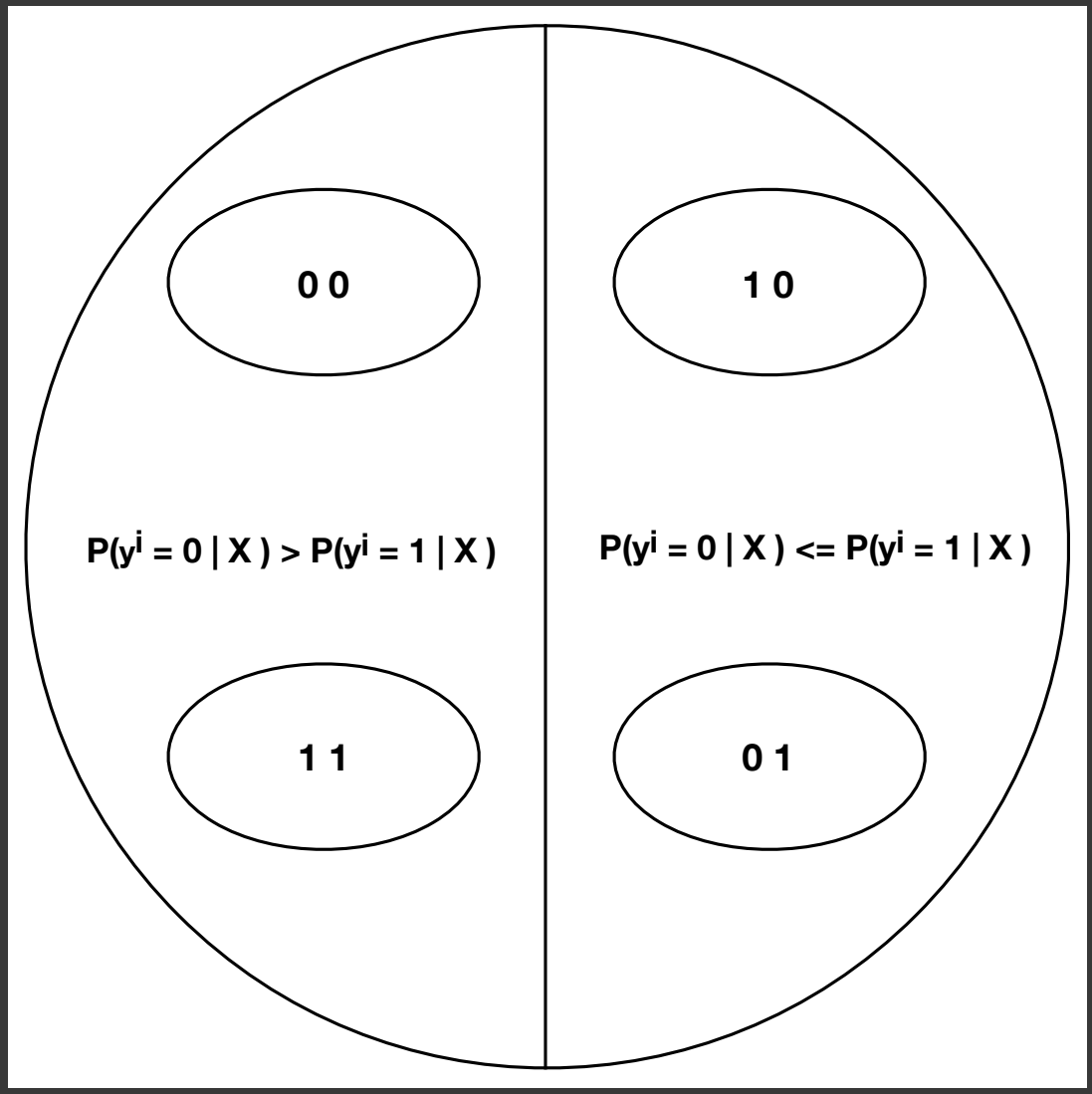
Input	Target
0 1 0 1 1 0 1 1 0 1 1 0	1
1 1 1 0 1 0 1 1 0 0 1 1	0
0 0 1 1 0 1 1 1 1 0 1 0	0
1 0 1 0 1 0 1 1 1 0 1 1	0
1 1 1 0 1 1 1 0 1 1 0 0	1
1 0 0 0 0 0 0 1 0 1 1 0	1
1 1 1 0 1 1 1 0 0 0 1 0	0
1 0 1 0 1 0 1 1 1 0 1 1	1
0 0 0 0 1 1 0 1 1 1 0 0	0
1 0 0 0 1 0 1 1 0 0 1 0	1

The pattern recognition problem



$$X = [x_1, x_2, \ldots, x_o] \qquad y = f(X)$$

The pattern recognition problem



$X = [x_1, x_2]$ $f(X) = (x_1 \wedge \neg x_2) V (\neg x_1 \wedge x_2)$

Conjunctive clauses for recognizing sub-patterns

Positive polarity

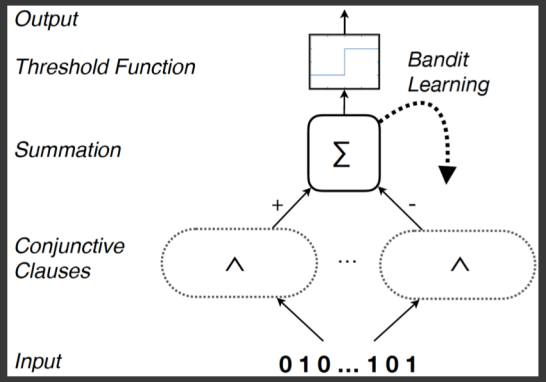
Non-negated variable

Negated variable

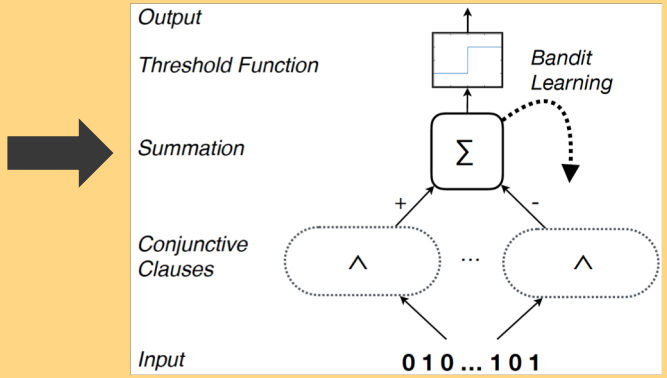
$$C_j^+(X) = x_1 \wedge \neg x_2$$

Negative polarity

$$C_j^-(X) = x_1 \wedge x_2$$

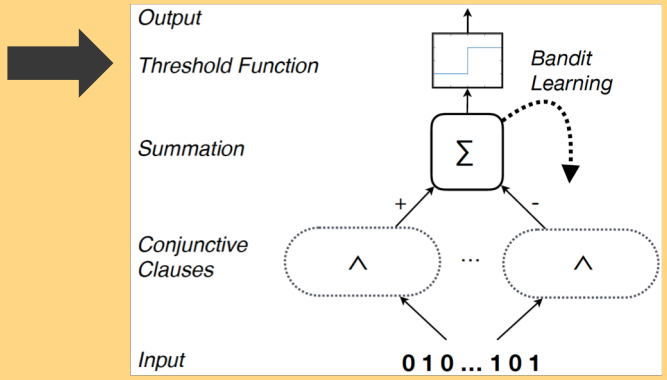


Adding up evidence from clauses



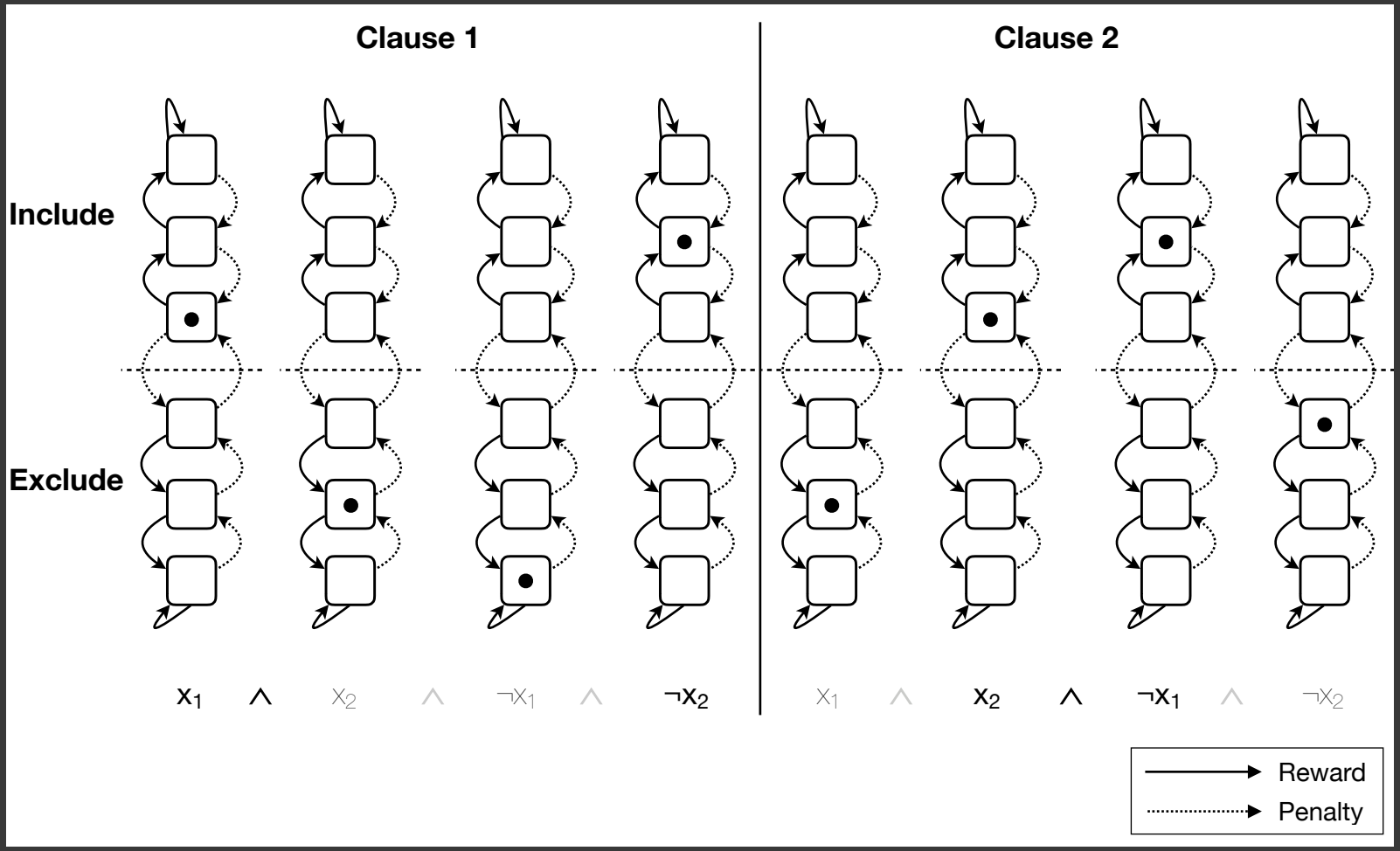
$$f_{\Sigma}(X) = \left(\sum_{j=1}^{m/2} C_j^{+}(X)\right) - \left(\sum_{j=1}^{m/2} C_j^{-}(X)\right)$$

A threshold function decides the final output



$$y = f_{\Sigma}(X) \geq 0$$

The conjunctive clauses are composed by teams of Tsetlin Automata

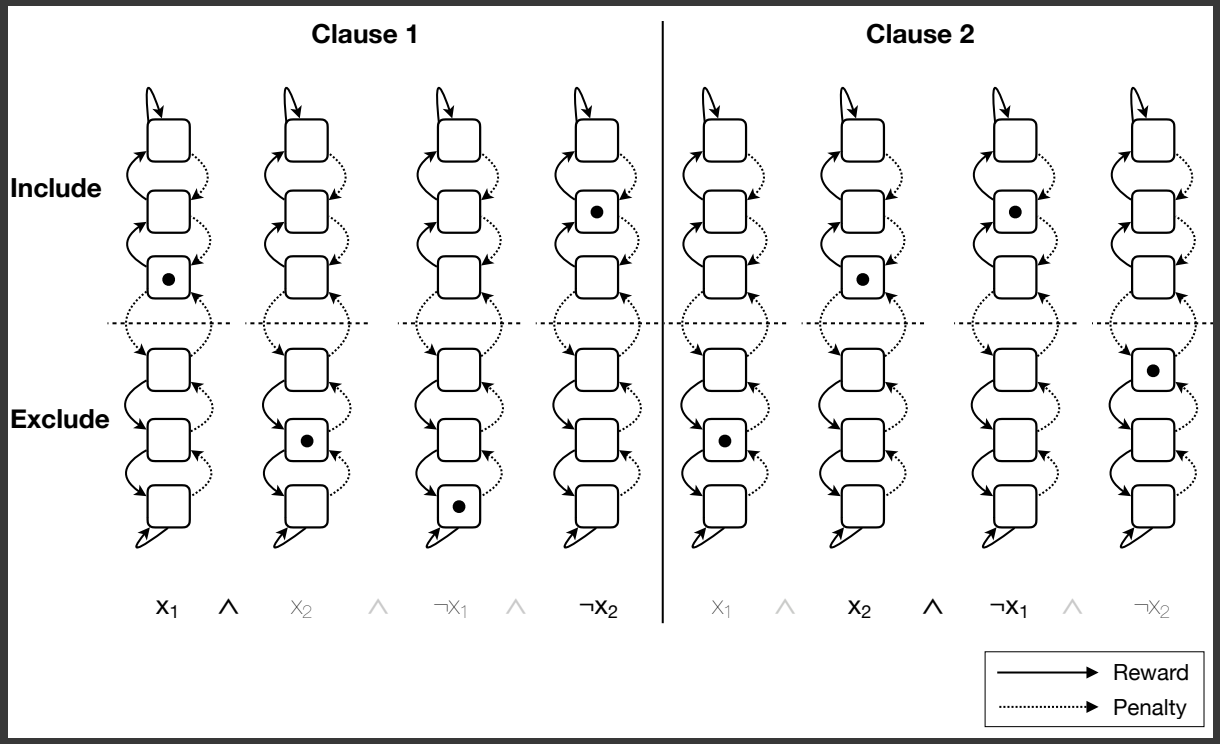


$C_1 = x_1 \wedge \neg x_2$ $C_2 = \neg x_1 \wedge x_2$

Exercise: Calculate output

Summation:

$$f_{\Sigma}(X) = \left(\sum_{j=1}^{m/2} C_j^{+}(X)\right) - \left(\sum_{j=1}^{m/2} C_j^{-}(X)\right)$$



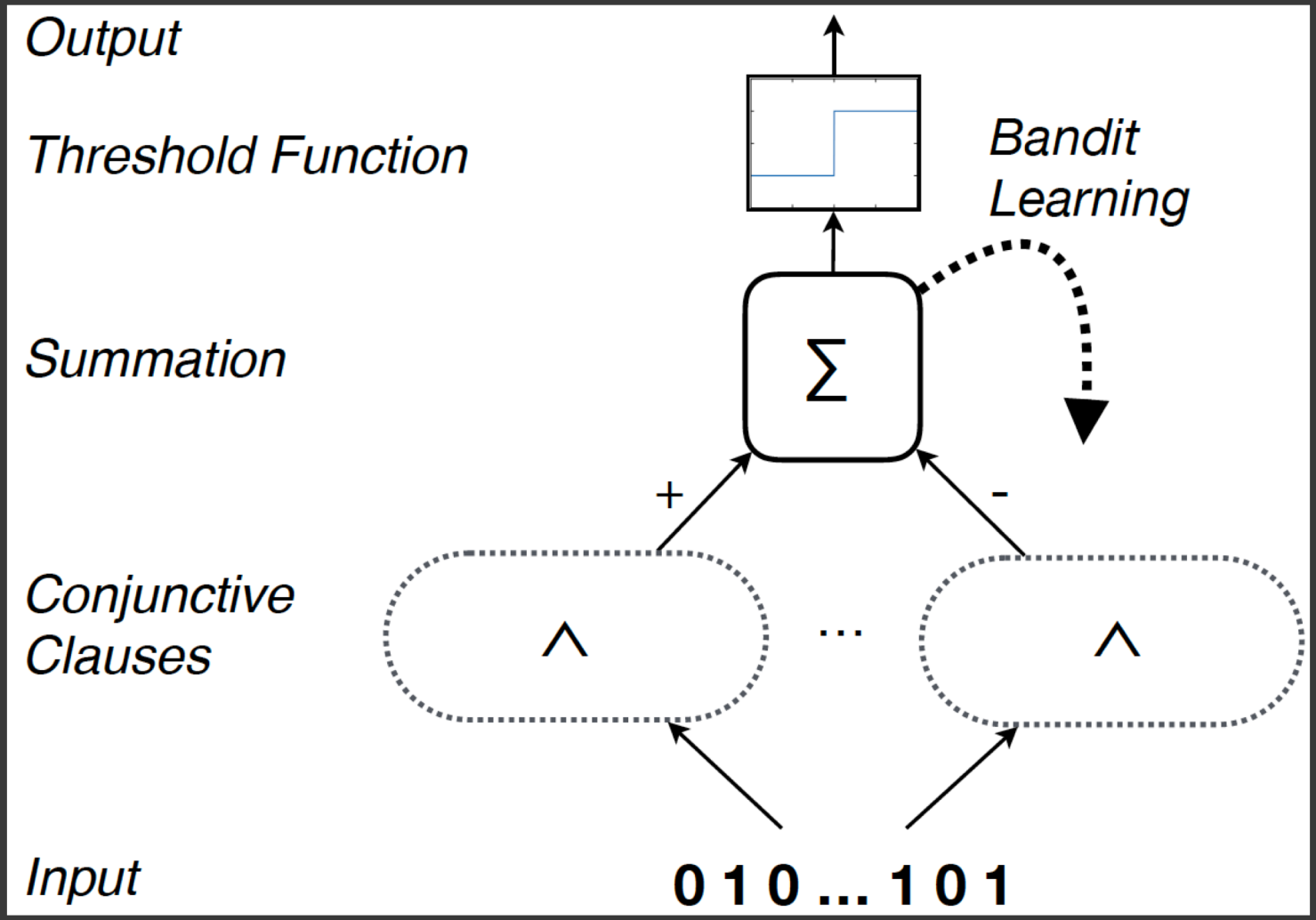
Clause evaluation:

Input:

$$X = [0, 1]$$

Learning with the Tsetlin Machine

The Tsetlin Machine

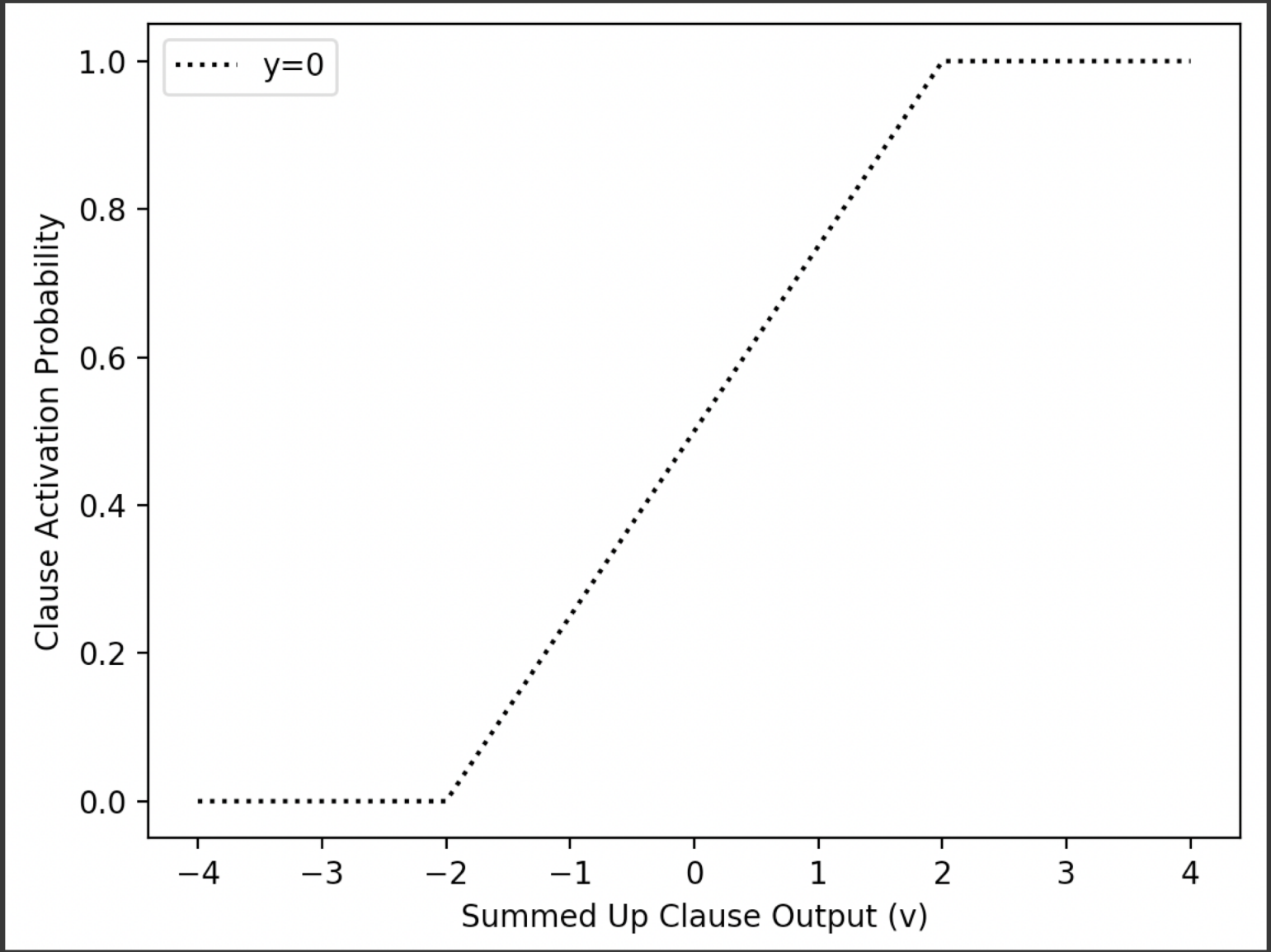


https://github.com/cair/TsetlinMachine

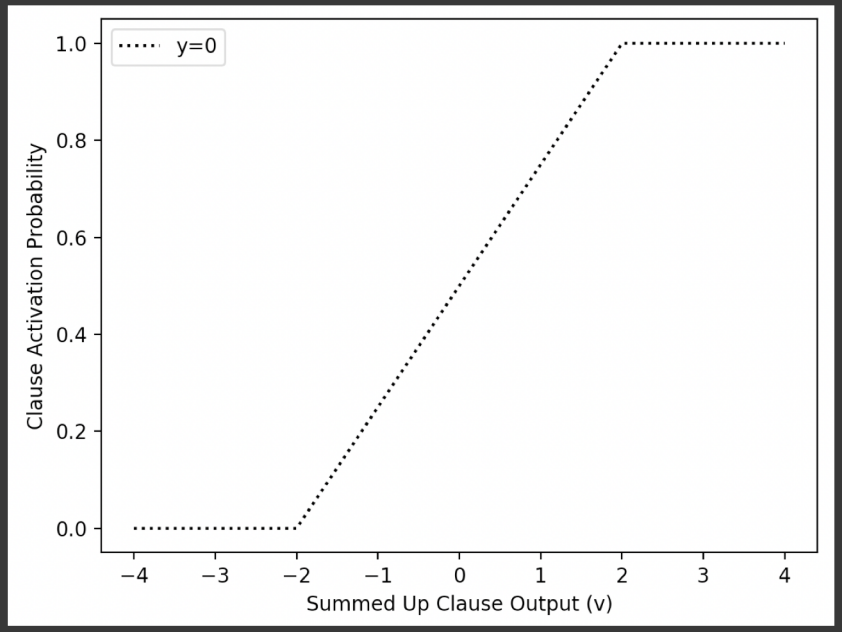
The Tsetlin Machine game

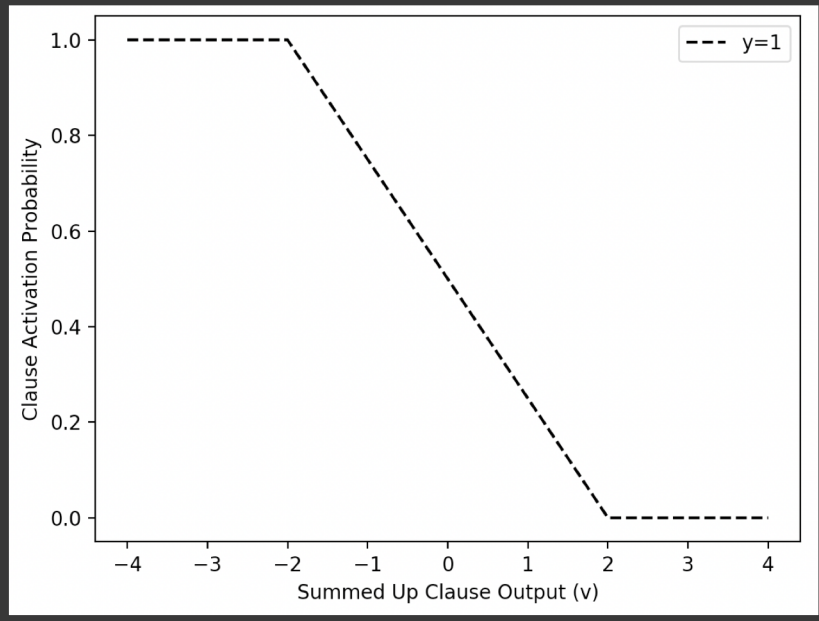
1. The arrival of a labelled object $(\hat{X}, \hat{y})$ signals the start of a new game round.
2. Each Tsetlin Automaton decides whether to *include* or *exclude* its designated literal, leading to a new configuration of clauses **C**
3. Each clause, $C_j \in$ **C**, is then evaluated with $\hat{X}$ as input.
4. Sum of votes v is calculated $v = f_{\Sigma}(X)$.
5. Target output $\hat{y}$ decides type of feedback.
6. Number of votes v is contrasted against a target value T, and a stochastic clause activation function decides which clauses are given feedback, decided by $\hat{y}$.
7. Each Tsetlin Automaton is independently and randomly given either Type I Feedback (for $\hat{y} = 1$) or Type II Feedback (for $\hat{y} = 0$).

Stochastic clause activation function



Stochastic clause activation function





Remark 1. Observe that the future returns of a clause are diminishing with the number of other clauses that capture the same sub-pattern. This is crucial for stimulating the Tsetlin Automata teams to distribute themselves across the critical sub-patterns.

Remark 2. A larger T (with a corresponding increase in the number of clauses) makes the learning more robust. This is because more clauses are involved in learning each specific sub-pattern, introducing an ensemble effect.

Type I feedback (true positive/false negative)

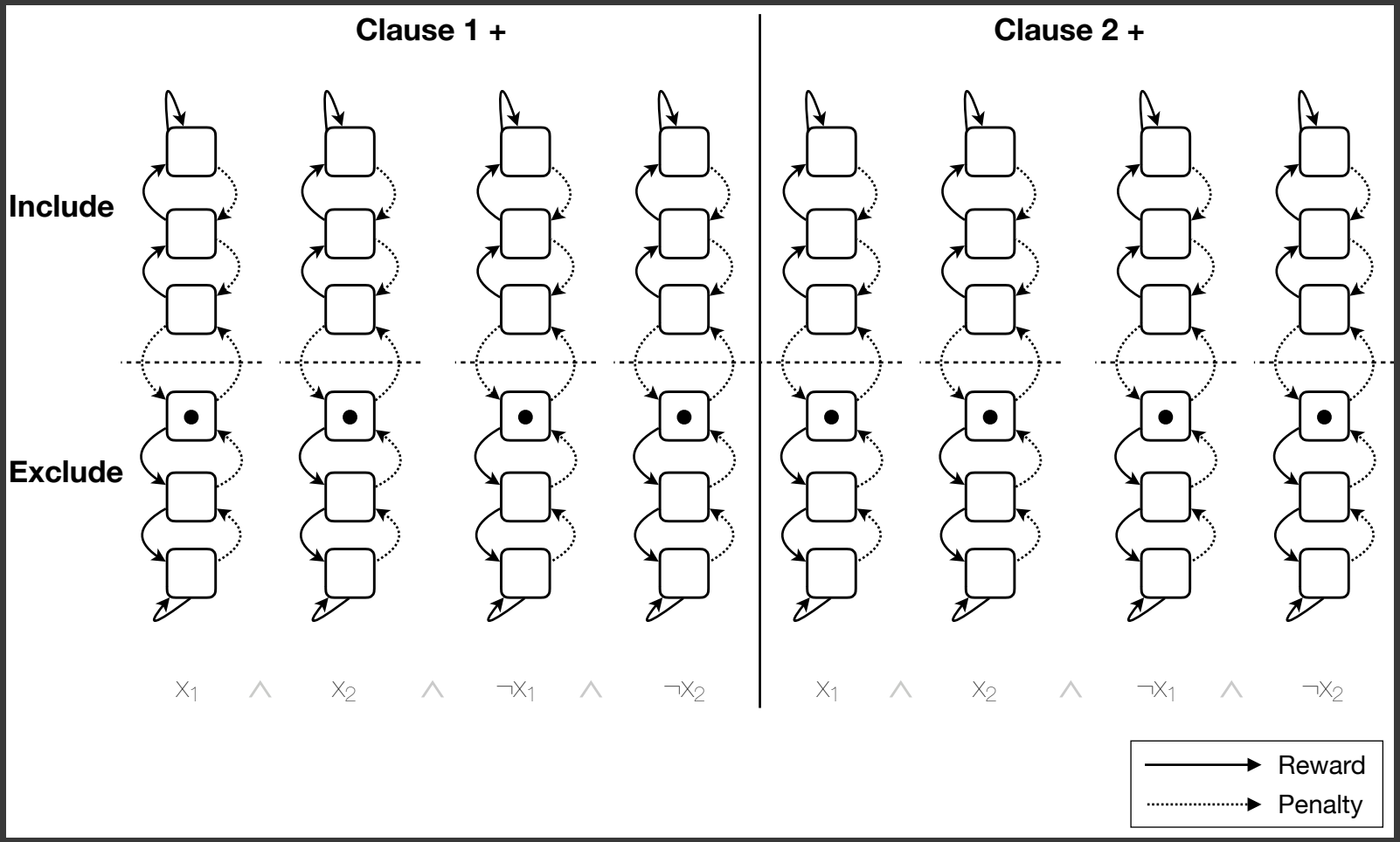
Target Clause evaluates to		1		0	
Target Literal evaluates to		1	0	1	0
Include literal	P (Reward)	$\frac{s-1}{s}$	NA	0	0
	P (Inaction)	$\frac{1}{s}$	NA	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	P (Penalty)	0	NA	$\frac{1}{s}$	$\frac{1}{s}$
Exclude literal	P (Reward)	0	$\frac{1}{s}$	$\frac{1}{s}$	$\frac{1}{s}$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

Type Ia Feedback – clause and literal true

Reinforces found sub-pattern, increasing precision by adding more literals to the clause

Type Ia Feedback – clause and literal true

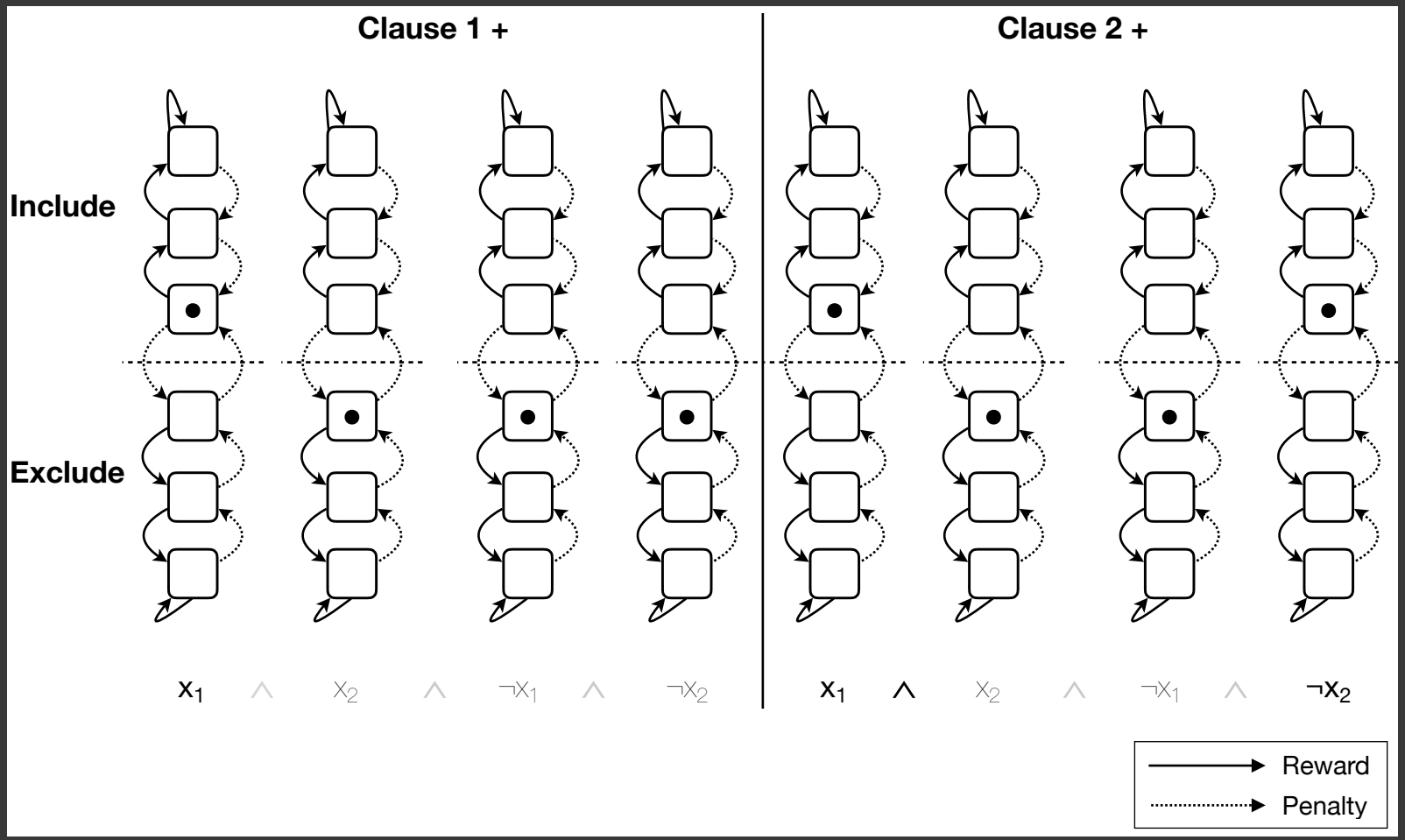
Init:



Input: $X = [1, 0], y = 1$

Type Ia Feedback – clause and literal true

Updated with Type Ia:



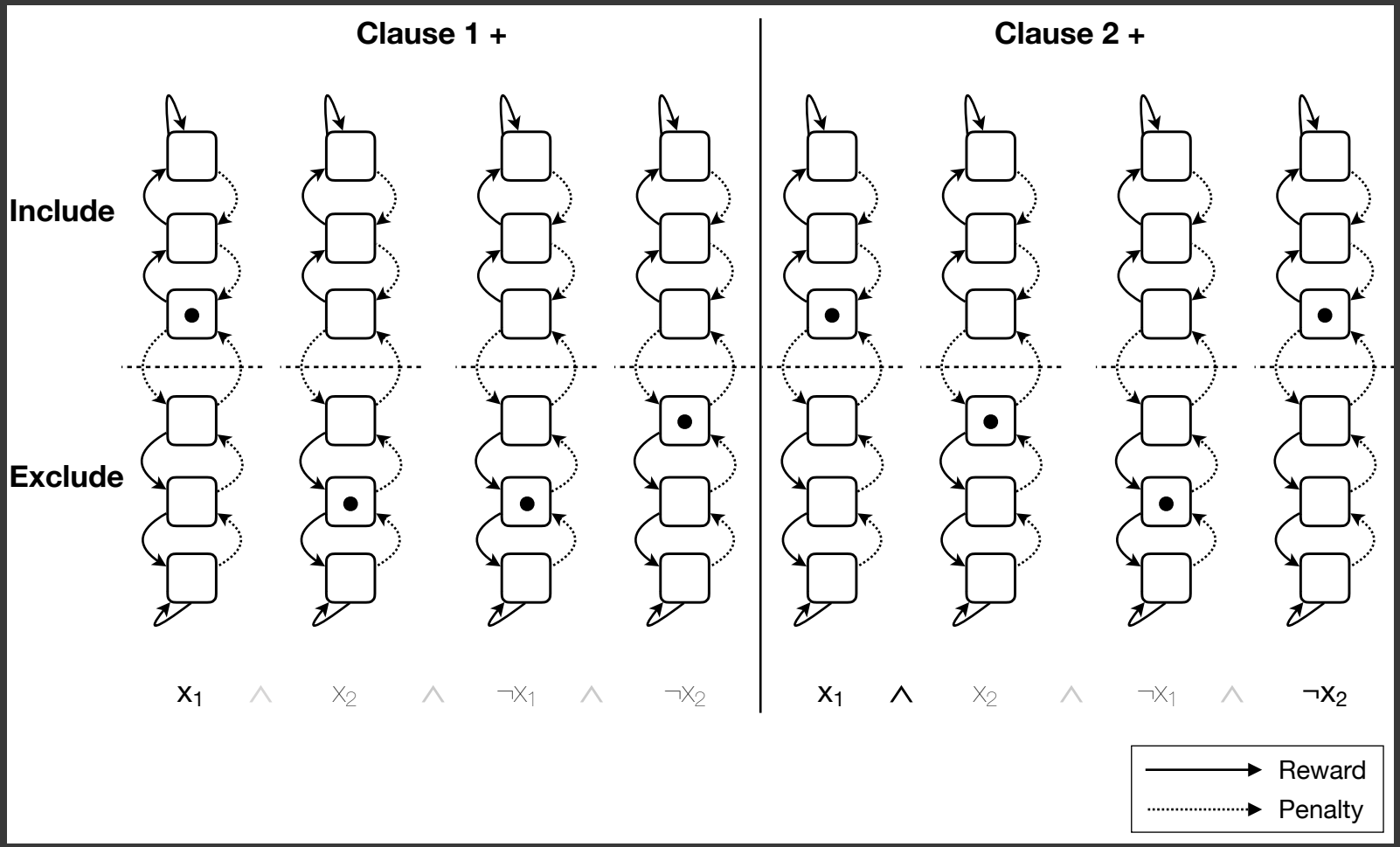
Input: $X = [1, 0], y = 1$

Type Ib Feedback – clause or literal false

Combats overfitting and erases patterns to make room for new, by reinforcing Exclude actions

Type Ib Feedback – clause or literal false

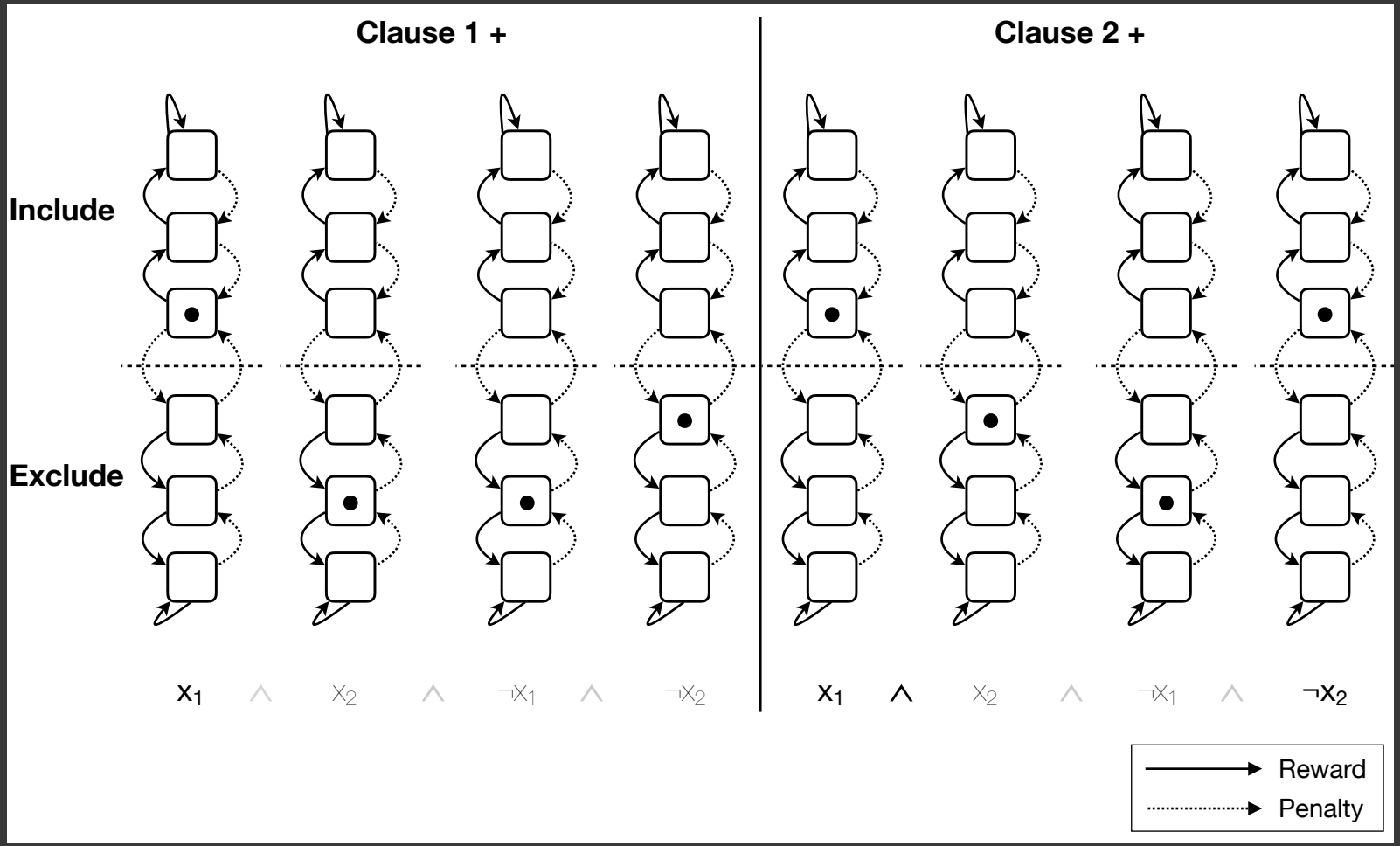
Updated with Type Ib:



Input: $X = [1, 0], y = 1$

Type Ib Feedback – clause or literal false

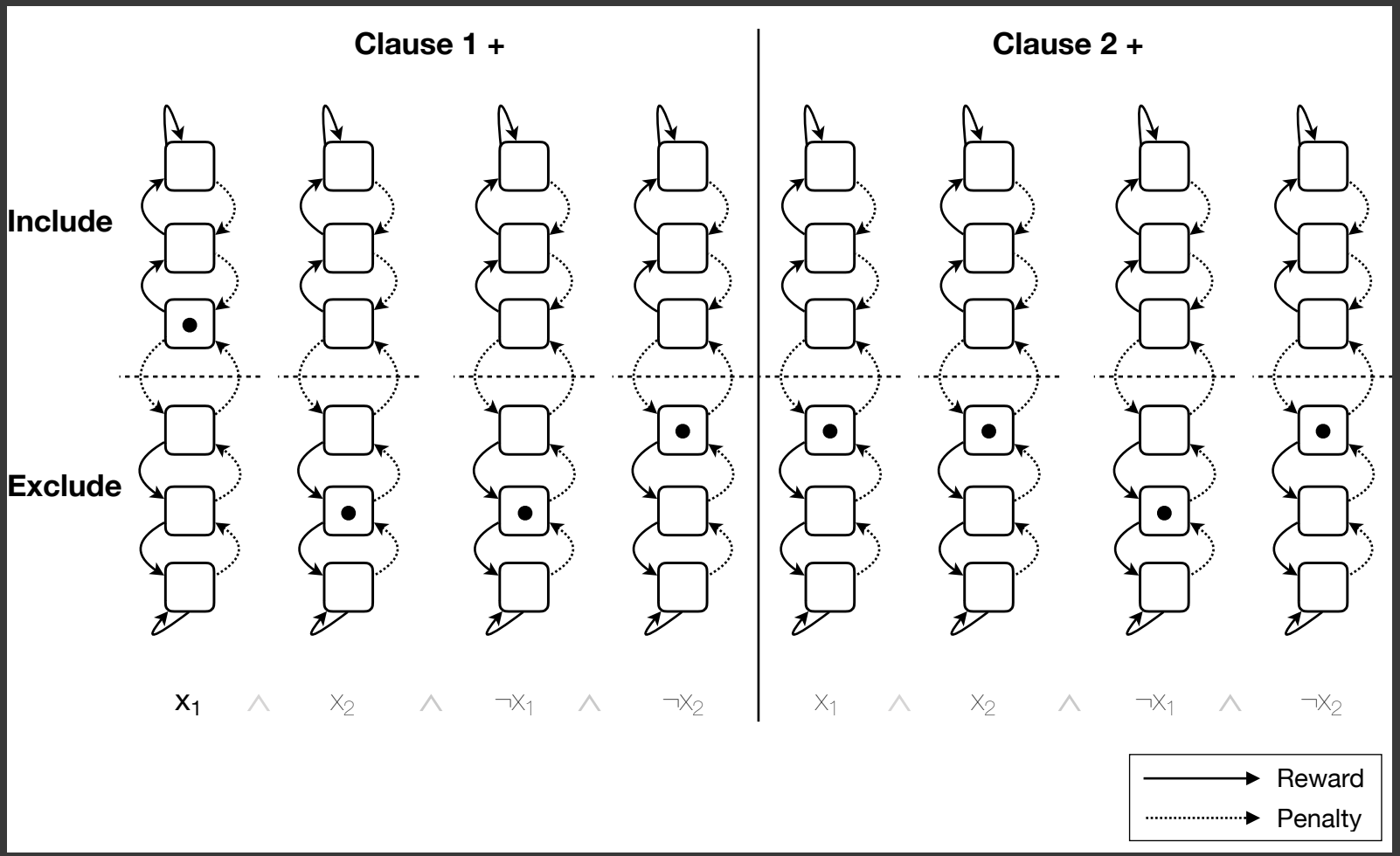
Init:



Input: $X = [0, 1], y = 1$

Type Ib Feedback – clause or literal false

Updated with Type Ib:



Input: $X = [0, 1], y = 1$

Type II Feedback – clause falsely outputs 1

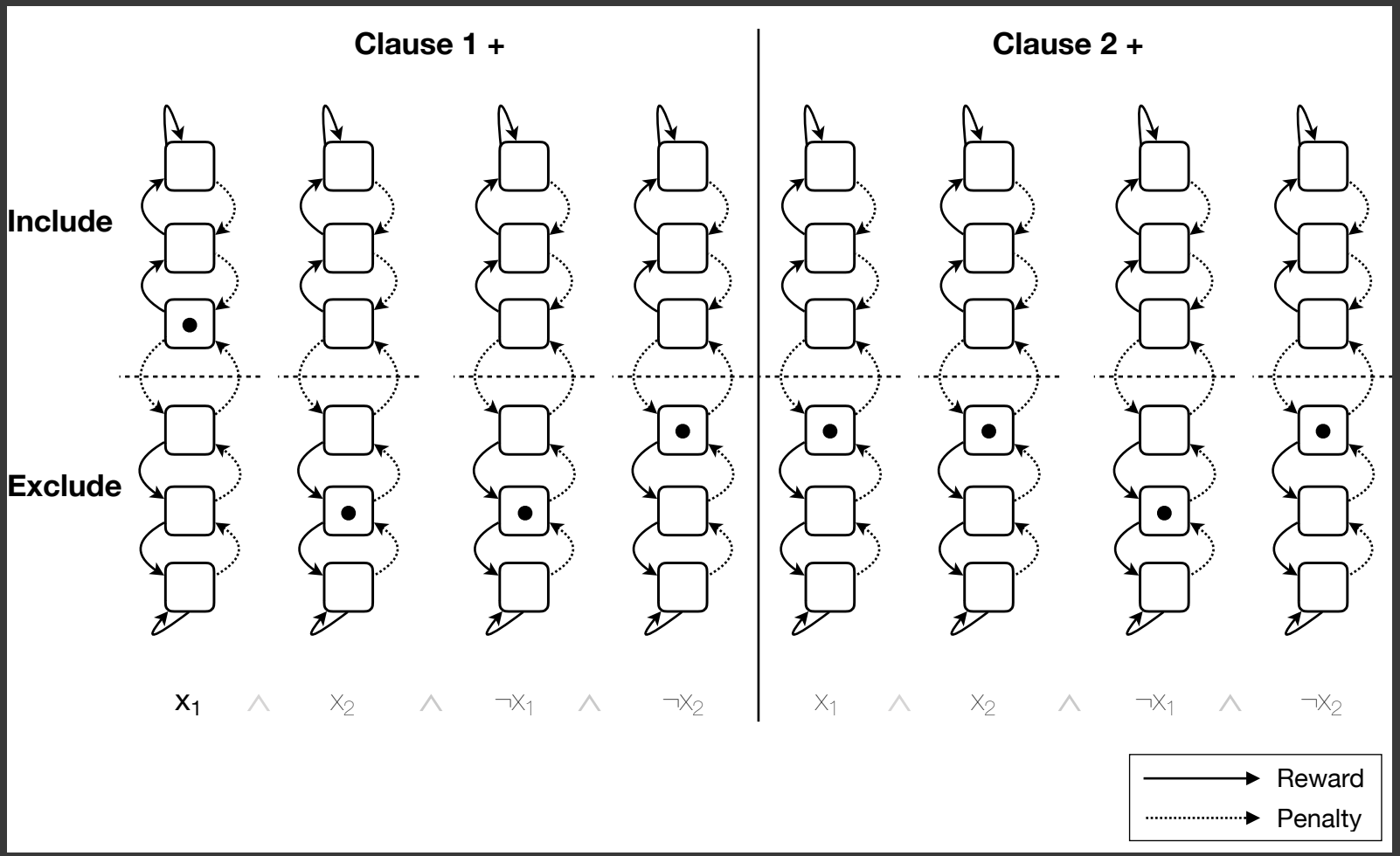
Increases discrimination power by including literals of value 0

Type II feedback (false positive)

Target Clause evaluates to		1		0	
Target Literal evaluates to		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

Type II Feedback – clause falsely outputs 1

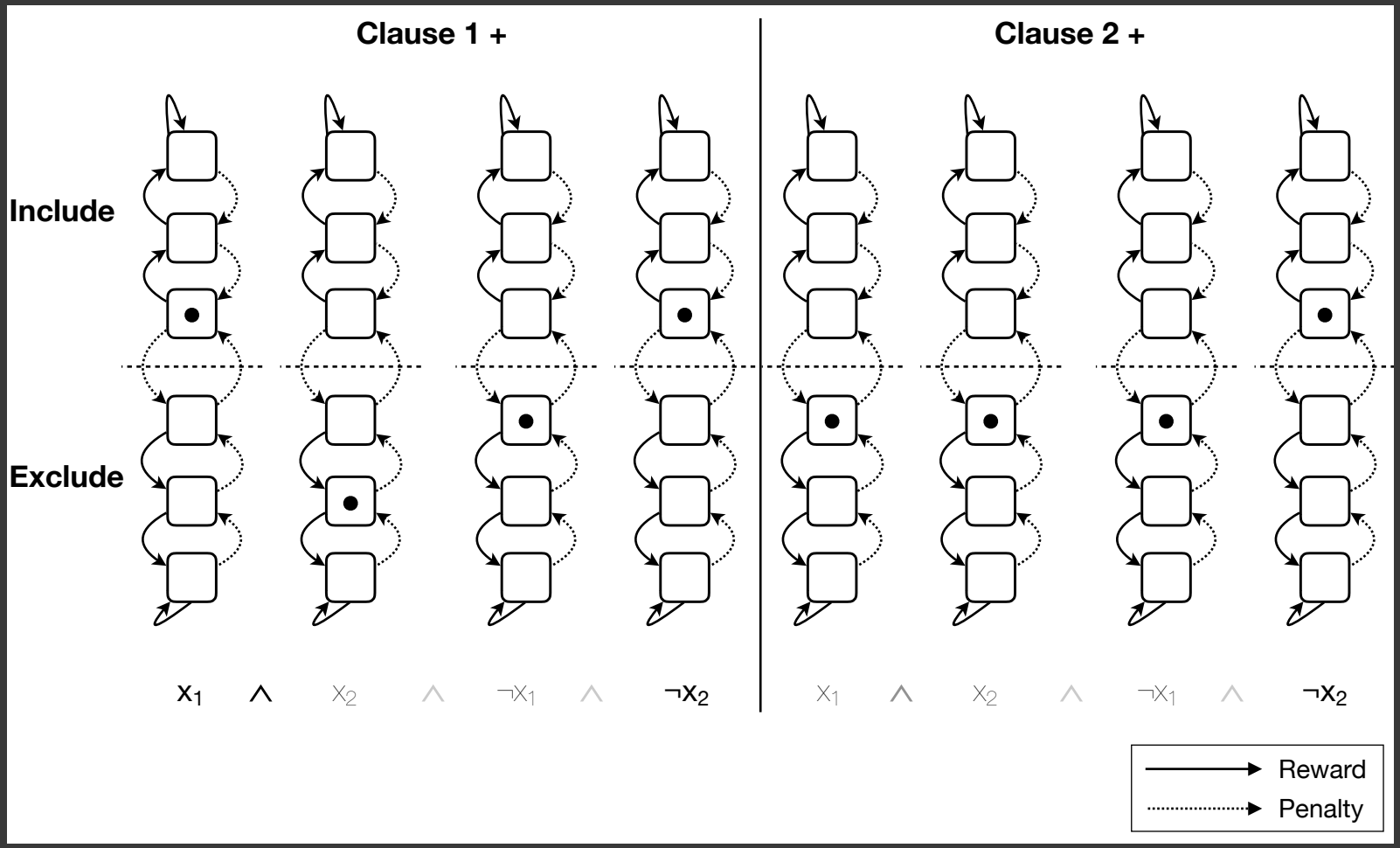
Init:



Input: $X = [1, 1], y = 0$

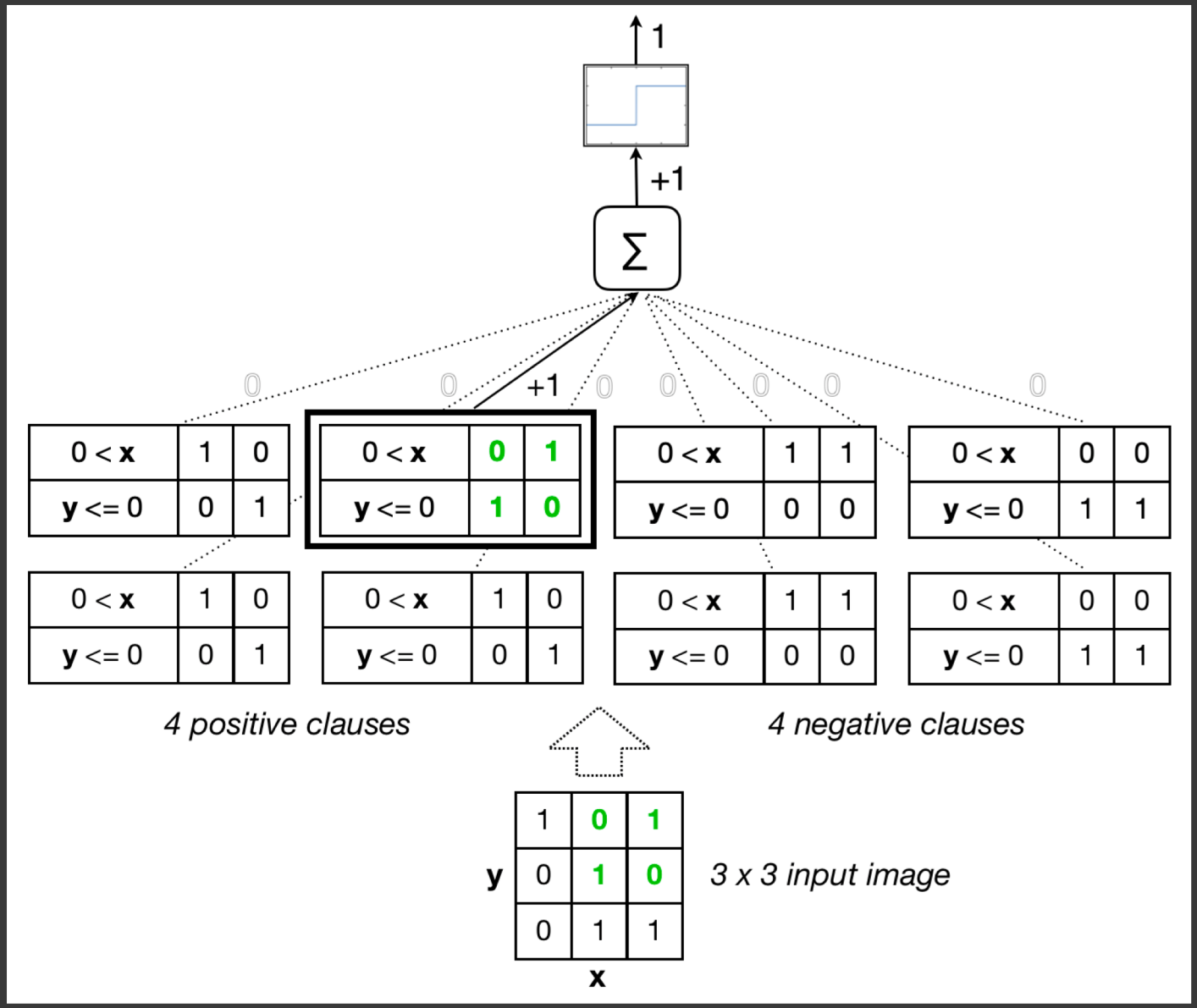
Type II Feedback – clause falsely outputs 1

Updated with Type II:

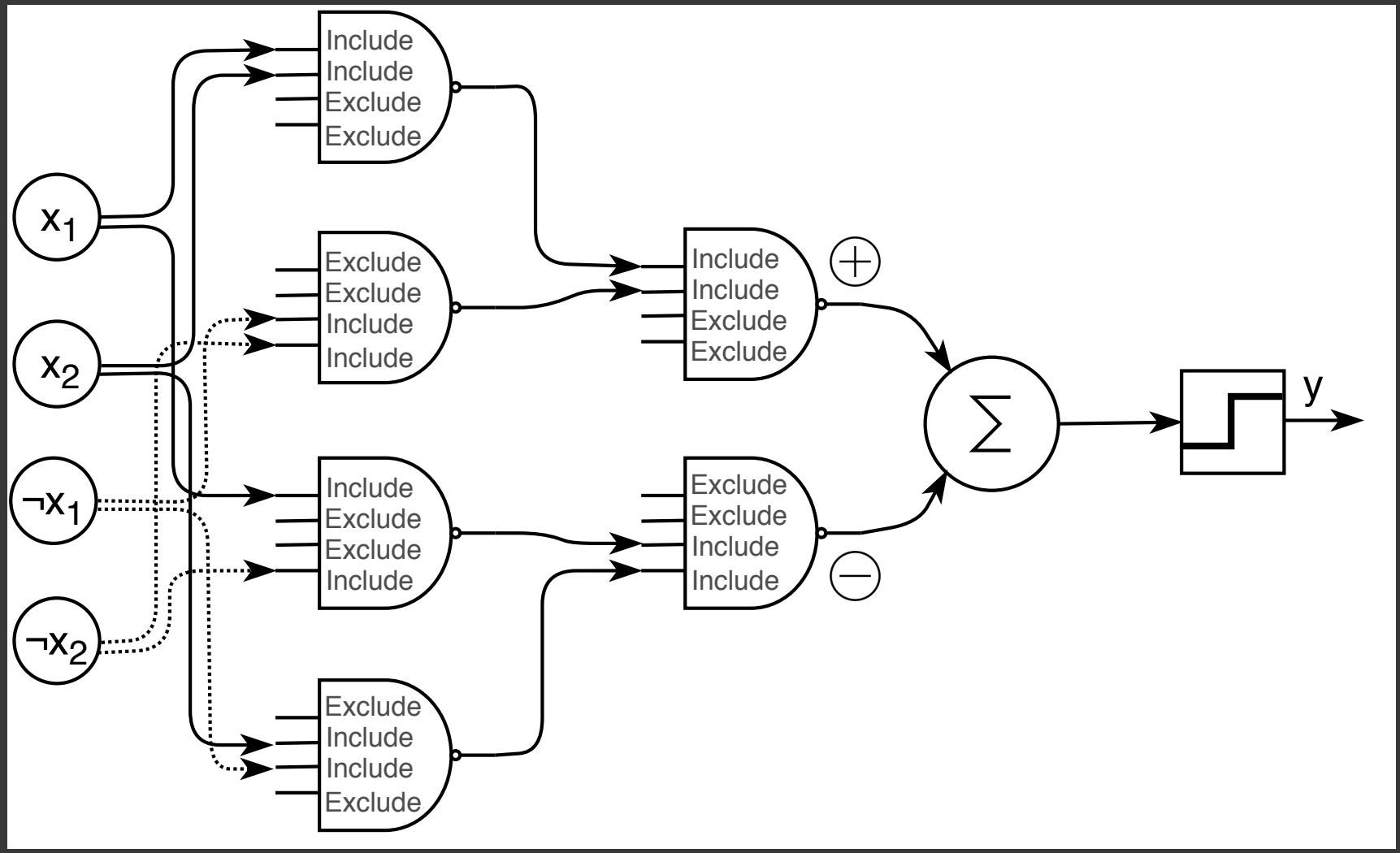


Input: $X = [1, 1], y = 0$

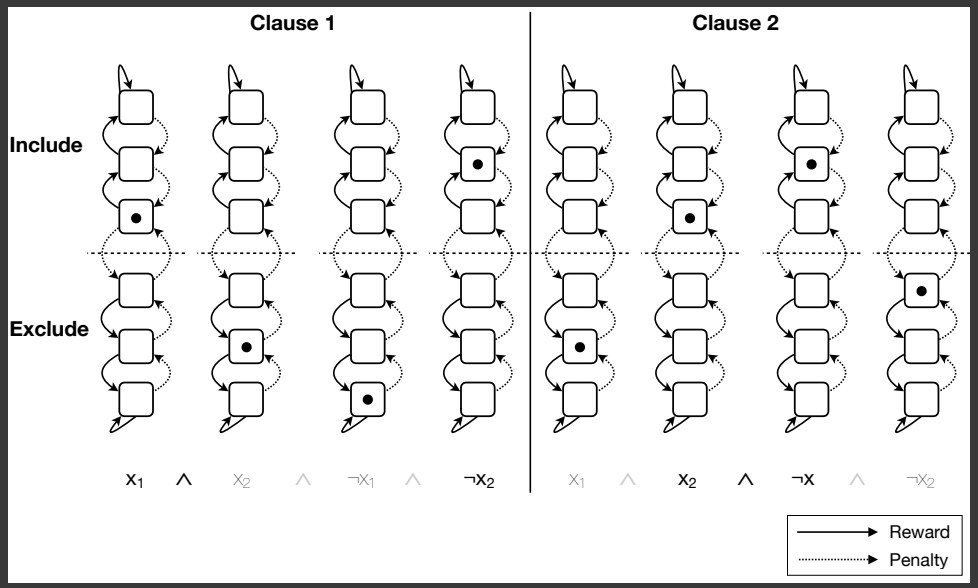
Convolutional Tsetlin Machine

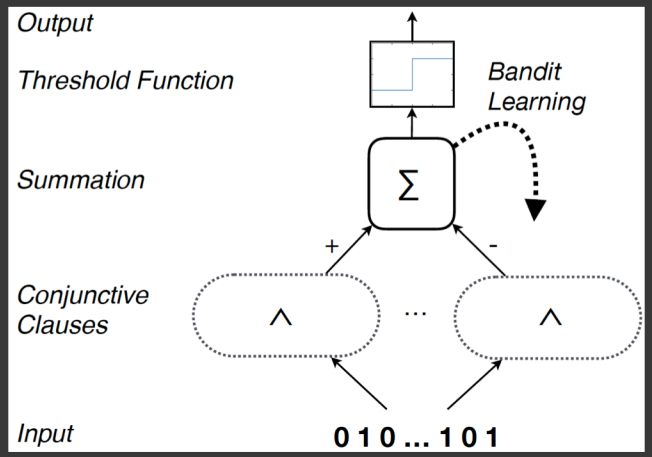


Multilayer Tsetlin Machine



Assignment 2





- Implement the above Tsetlin Machine architecture
- Evaluate learning accuracy using three different datasets: OR, AND, XOR
- Display the clauses produced
- Optional: Add noise by randomly inverting output in the datasets. Experiment with different hyperparameter values T and s

Assignment 2

Clause activation function for y = 1 (Feedback Type I):

Clause output sum	0	1	2
Activation probability	100.0	0.0	0.0

Clause activation function for y = 0 (Feedback Type II):

Clause output sum	0	1	2
Activation probability	0.0	100.0	100.0