Chapter 2

Classification

In Chapter 1, you learned how to implement a Tsetlin machine for classification tasks, obtaining intuition on how it works. The goal of Chapter 2 is to give you a deeper understanding. Through mathematical analysis, you discover the exact nature of Tsetlin machine learning, including a mathematical description of the learning outcome. The chapter is optional, and you can proceed to Chapter 3 and regression if you like to focus on building intuitive algorithm understanding.

In this chapter, you first study single-literal learning in Section 2.1. Single-literal learning is a so-called *stochastic process*. Specifying this process, you obtain simple formulas that describe the learning outcome exactly. The learning outcome reveals how the Tsetlin machine picks up frequent literals, emphasizing literals that distinguish between objects of different classes. You will also see that the learning gets more accurate by increasing memory depth.

In Section 2.2, you follow the same strategy for rules with two literals. However, this time, you get a more sophisticated stochastic process that also captures how literals interact. The well-known Prisoner's Dilemma from game theory casts light on this interaction. The learning outcome shows how the Tsetlin machine selects the best literals out of many and combines multiple for increased precision.

Finally, you investigate two-rule learning in Section 2.3. The stochastic process now covers rule interaction, which emerges from literal interaction.

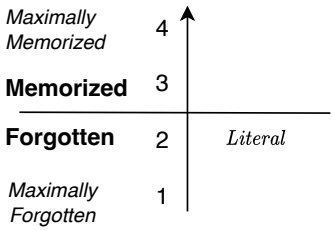

Figure 2.1: A rule memory of depth 4 with a single literal in position 2.

It is the Voting Margin that governs the interaction between rules. You uncover how the Tsetlin machine avoids sub-optimal configurations — so-called local optima — producing optimal ones instead.

In all these cases, you will see that the outcome of Tsetlin machine learning is not a single solution. Instead, it is a probability distribution over solutions. The Tsetlin machine never stands still, visiting all the candidate solutions with continuous learning.

2.1 Single-Literal Learning

Recall from Chapter 1 how the Tsetlin machine keeps literals in graded memory, ranging from 1 (Maximally Forgotten) to 10 (Maximally Memorized). You can vary the value for Maximally Memorized to create memories of different depths. For example, the memory in Figure 2.1 is of depth four and contains a single literal in position two. As you know from Chapter 1, Recognize, Erase, and Reject feedback change the literal's location in memory during learning.[1] The question is: *What is the outcome of this learning process?*

[1]Please note that I have renamed Type I and Type II Feedback from the first version of Chapter 1 as Recognize, Erase, and Reject Feedback. The reason is to make the feedback types easier to remember. A new version of Chapter 1 is now available.

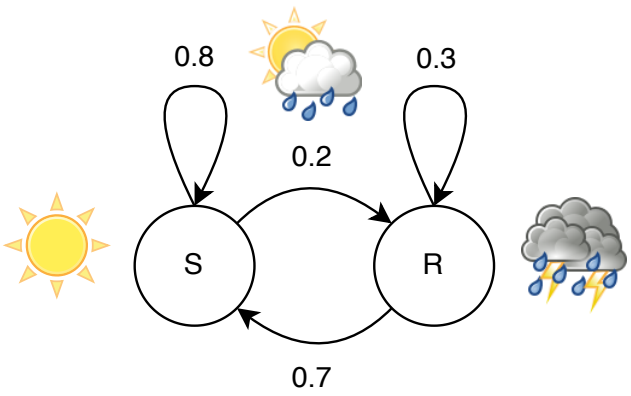

Figure 2.2: Transition graph of a weather process.

Markov Chain

You find the answer to the question using a *Markov chain*. In brief, a Markov chain describes a sequence of states, illustrated in Figure 2.2. The figure depicts how the weather changes day-to-day, from *Sunny* to *Sunny*, *Sunny* to *Rainy*, *Rainy* to *Rainy*, and *Rainy* to *Sunny*.

Transition Graph. A graph describes how the weather changes. It consists of *nodes* and *edges*:

- **Nodes.** The nodes are the possible states, S or R, that is, *Sunny* or *Rainy* weather.
- **Edges.** The edges are the state transitions: $S \rightarrow S$, $S \rightarrow R$, $R \rightarrow R$, and $R \rightarrow S$. Each edge has a probability, which is the probability of the transition. So, the weather remains *Sunny* (S) with probability 0.8 and transitions to *Rainy* (R) with probability 0.2.

This is the *transition graph* of the Markov chain. A sequence of states could look like this: $SSRS$. Two days are *Sunny*, followed by a *Rainy* day, followed by a *Sunny* day. The probability of these transitions are $0.8 \cdot 0.2 \cdot 0.7 = 0.112$.

Markov Property. The defining property of a Markov chain is that the probability of a state transition only depends on the current state. For the weather, the Markov property says that tomorrow's weather only depends on today's.

Stationary Distribution. To discover more about a Markov chain, you can run it for a long time. Then you find the probability of a *Sunny* or *Rainy* day happening in general. These probabilities are the *stationary distribution* of the chain — the state probabilities that the Markov chain stabilizes on and that remain unchanged as time progresses. Mathematically, you get the stationary distribution by solving the following equations:

$$P(S) = 0.8 \cdot P(S) + 0.7 \cdot P(R) \quad (2.1)$$
$$P(R) = 0.2 \cdot P(S) + 0.3 \cdot P(R) \quad (2.2)$$
$$1 = P(S) + P(R). \quad (2.3)$$

Here, $P(S)$ is the probability of *Sunny* weather and $P(R)$ is the probability of *Rainy* weather. Referring to Figure 2.2, notice the following:

- The right-hand side of Eqn. 2.1 calculates the probability of ending up in state S. You end up in S if you already are in state S and stay there: $0.8 \cdot P(S)$. You also end up in S if you are in state R, but transition to S: $0.7 \cdot P(R)$. The state probability $P(S)$ also appears on the left-hand side. *You seek the state probability that does not change after a transition.*
- Similarly, Eqn. 2.2 says that state probability $P(R)$ also must remain unaffected by the transition.
- Finally, Eqn. 2.3 captures that you always stay either in state S or R. Hence, the probabilities sum to 1.

Solving the equations, you get:

$$P(S) \approx 0.78 \quad (2.4)$$
$$P(R) \approx 0.22. \quad (2.5)$$

In other words, the probability of *Sunny* weather on any day becomes 0.78 after the chain has progressed for some time. The probability of *Rainy* weather becomes 0.22.

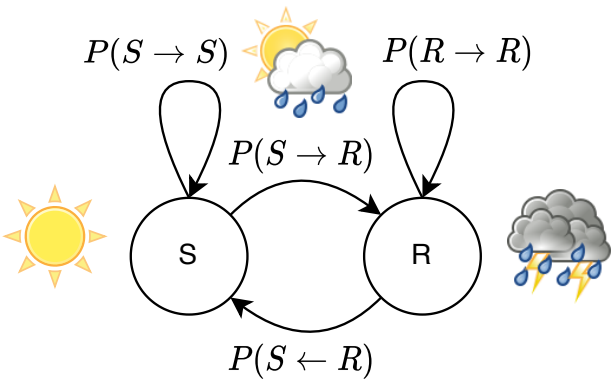

Figure 2.3: Generic transition graph of the weather process.

Stationary Distribution with Symbols. It is also possible to find the stationary distribution when you do not know the transition probabilities. You simply replace them with symbols. Figure 2.3 depicts the symbolic version of the weather process. The probability 0.8 of transitioning from S to S is replaced by $P(S \rightarrow S)$, for instance. Solving the equations using symbols gives you more insight into the weather. You obtain the probability of *Sunny* and *Rainy* weather for *any* transition probabilities:

$$P(S) = \frac{P(R \rightarrow S)}{P(R \rightarrow S) - P(S \rightarrow S) + 1} \tag{2.6}$$

$$P(R) = \frac{P(S \rightarrow R)}{P(R \rightarrow S) - P(S \rightarrow S) + 1}. \tag{2.7}$$

Markov Chain of Single Literal Learning

In the same way that you determine the probability of *Sunny* and *Rainy* weather, you can find the learning outcome of a Tsetlin machine. To do this, you start with composing the Markov chain of single-literal learning. The resulting learning outcome equations give you important insight into the Tsetlin machine.

The Literal States. First, you need the states, like *Sunny* and *Rainy* weather. In brief, the literal's position in memory becomes its state. So, the memory in Figure 2.4 of depth four gives four states. These states are

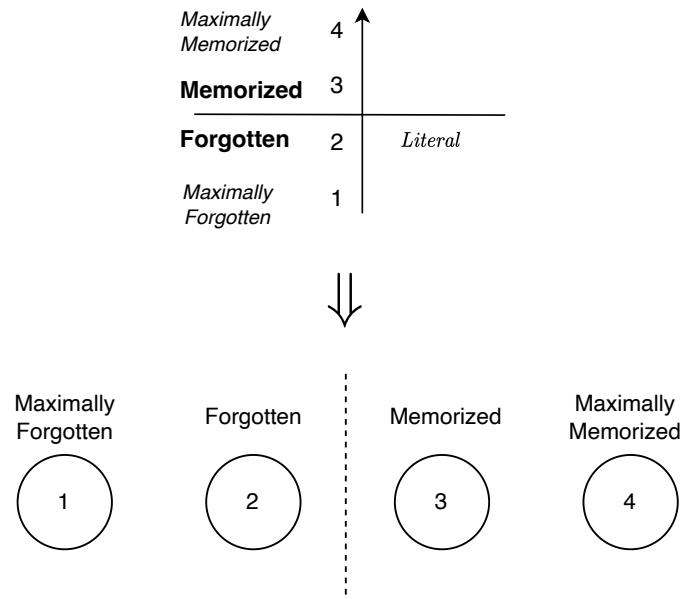

Figure 2.4: A four-state Literal Automaton.

numbered 1 (Maximally Forgotten) to 2 (Forgotten) and then 3 (Memorized) to 4 (Maximally Memorized). I will, from now on, refer to a literal with memory as a *Literal Automaton* to highlight that it is a so-called *finite-state automaton*.

Class. Next, you need the classes. Instead of recognizing a specific class, like *Car* or *Recurrence* from Chapter 1, you now use the generic class Y. Further, let $\overline{Y}$ mean any class other than Y.

Literal Truth Value. You also need symbols for the truth values of the literal. Instead of saying that for instance *Four Wheels* is *True*, use the symbol L. Conversely, use $\overline{L}$ to tell that *Four Wheels* is *False*.

Training Examples. Together with the class, a single literal provides four different kinds of observations:

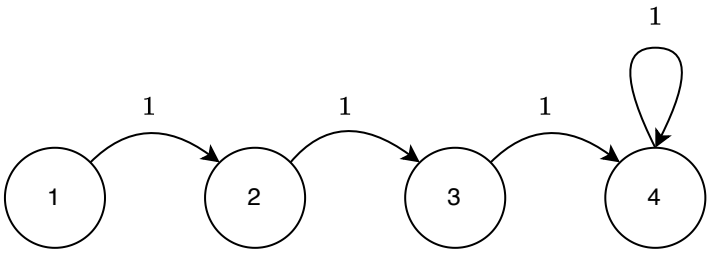

Figure 2.5: Recognize Feedback with four-state Literal Automaton and Memorize Value 1.0.

1. Observation (L, Y) means the literal is *True* and the class is Y.
2. Observation $(\overline{L}, Y)$ means the literal is *False* and the class is Y.
3. Observation $(L, \overline{Y})$ means the literal is *True* and the class is $\overline{Y}$.
4. Observation $(\overline{L}, \overline{Y})$ means the literal is *False* and the class is $\overline{Y}$.

Each kind of observation gives different memory updates. Recall from Chapter 1 how Recognize, Erase, and Reject Feedback operate. You can then proceed to build the Markov chain.

Recognize Feedback. Recognize Feedback happens when you observe (L, Y). Figure 2.5 shows the transitions of Recognize Feedback. Notice how each transition increases the literal's position in memory with probability 1.0 (up to Maximally Memorized), meaning that the transitions always happen. From now on, always use Memorize Value 1.0 for Recognize Feedback.

Erase Feedback. Erase Feedback happens when you observe $(\overline{L}, Y)$ and it decreases the literal's position in memory randomly according to the Forget Value. Use Forget Value $\frac{1}{s}$ from now on, as typical in the research literature. The symbol s refers to *specificity* — increasing s makes the rules more specific. Erase Feedback then decreases the literal's position with probability $\frac{1}{s}$, unless already in State 1 (Maximally Forgotten). This means that the literal stays in place with probability 1 for State 1, and with

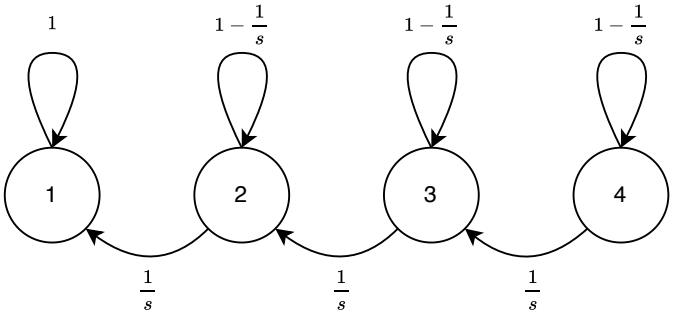

Figure 2.6: Erase Feedback with four-state Literal Automaton and Forget Value $\frac{1}{s}$.

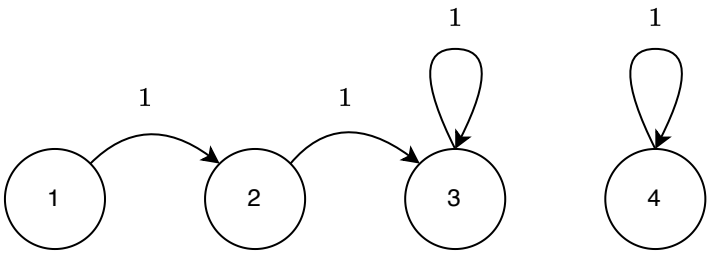

Figure 2.7: Reject Feedback with four-state Literal Automaton.

probability $1 - \frac{1}{s}$ for States 2 to 4. Figure 2.6 shows the Erase Feedback transitions.

Reject Feedback. Reject Feedback occurs whenever you observe $(\overline{L}, \overline{Y})$. If the literal is on the *Forgotten* side of memory, increase its position. Otherwise, let it stay in place. Figure 2.7 shows the resulting transitions for the four-state Literal Automaton. The literal is on the *Forgotten* side in State 1 and State 2, hence its position increases only in those two states.

Feedback Combined. You now have three different transition graphs for the Markov chain you build. You use the Recognize Feedback transitions for observation (L, Y), the Erase Feedback transitions for observation $(\overline{L}, Y)$, and the Reject Feedback transitions for observation $(\overline{L}, \overline{Y})$. Finally, nothing happens when you observe $(L, \overline{Y})$. This is the No Feedback case.

As you see, each of these four different kinds of observations give different transitions. You therefore need to know the probability of each happening. Then you can build the full Markov chain for single-literal learning. In other words, you need to know: $P(LY)$, $P(\overline{L}Y)$, $P(L\overline{Y})$, and $P(\overline{L}\overline{Y})$. From the standard definition of conditional probabilities, you can rewrite these as: $P(L|Y)P(Y)$, $P(\overline{L}|Y)P(Y)$, $P(L|\overline{Y})P(\overline{Y})$, and $P(\overline{L}|\overline{Y})P(\overline{Y})$, respectively. With the latter information in place, you can now obtain the probability of transitioning from State i to State j, denoted $P(i \rightarrow j)$:

$$\begin{aligned} P(i \rightarrow j) &= P(i \rightarrow j|\text{“Recognize”})P(\text{“Recognize”}) && (2.8)\\ &+ P(i \rightarrow j|\text{“Erase”})P(\text{“Erase”}) && (2.9)\\ &+ P(i \rightarrow j|\text{“Reject”})P(\text{“Reject”}) && (2.10)\\ &+ P(i \rightarrow j|\text{“No Feedback)}P(\text{“No Feedback”}). && (2.11) \end{aligned}$$

Since each feedback type triggers from its own observation, you can rewrite the above formula as follows:

$$\begin{aligned} P(i \rightarrow j) &= P(i \rightarrow j|LY)P(L|Y)P(Y) && (2.12)\\ &+ P(i \rightarrow j|\overline{L}Y)P(\overline{L}|Y)P(Y) && (2.13)\\ &+ P(i \rightarrow j|\overline{L}\overline{Y})P(\overline{L}|\overline{Y})P(\overline{Y}) && (2.14)\\ &+ P(i \rightarrow j|L\overline{Y})P(L|\overline{Y})P(\overline{Y}). && (2.15) \end{aligned}$$

Using this formula on each Literal Automaton transition, you obtain the Markov chain for single-literal learning in Figure 2.8. As an example, you

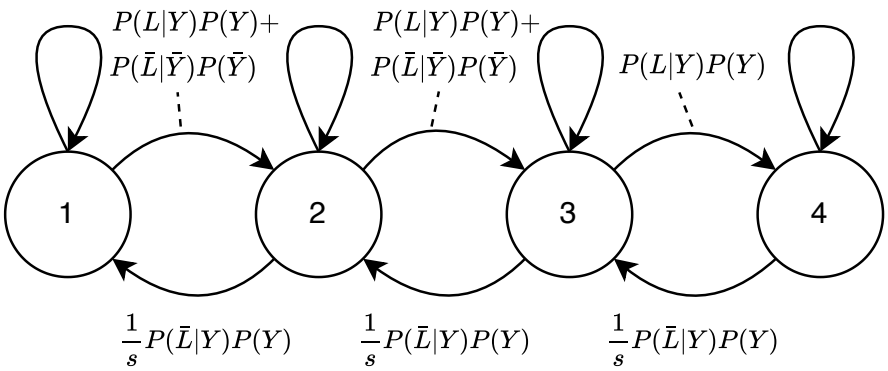

Figure 2.8: Transition graph of four-state Literal Automaton.

get $P(1 \rightarrow 2)$ from:

$$\begin{aligned} P(1 \rightarrow 2) &= P(1 \rightarrow 2|LY)P(L|Y)P(Y) && (2.16)\\ &+ P(1 \rightarrow 2|\overline{L}Y)P(\overline{L}|Y)P(Y) && (2.17)\\ &+ P(1 \rightarrow 2|\overline{L}\overline{Y})P(\overline{L}|\overline{Y})P(\overline{Y}) && (2.18)\\ &+ P(1 \rightarrow 2|L\overline{Y})P(L|\overline{Y})P(\overline{Y}). && (2.19) \end{aligned}$$

Then you substitute with the actual transition probabilities of each feedback type:

$$\begin{aligned} P(1 \rightarrow 2) &= 1 \cdot P(L|Y)P(Y) && (2.20)\\ &+ 0 \cdot P(\overline{L}|Y)P(Y) && (2.21)\\ &+ 1 \cdot P(\overline{L}|\overline{Y})P(\overline{Y}) && (2.22)\\ &+ 0 \cdot P(L|\overline{Y})P(\overline{Y}). && (2.23) \end{aligned}$$

Notice that one transition probability is left out per state in the figure. To save space, the figure omits the probability of staying in place, i.e., $P(i \rightarrow i)$. This probability is simply 1 minus the sum of the outbound transition probabilities. For instance, the probability of staying in State 1 is

$$P(1 \rightarrow 1) = 1 - (P(L|Y)P(Y) + P(\overline{L}|\overline{Y})P(\overline{Y})). \quad (2.24)$$

Single-Literal Transition Matrix

With the large number of transition probabilities, organizing them in a matrix $\mathbf{M}$ is helpful. Place each transition probability $P(i \rightarrow j)$ in the row i and column j entry of the matrix. The formulas quickly become lengthy, so, to save space, use shorthand notation for the probabilities. Use π_i to denote the probability of being in State i, i.e., $\pi_i = P(i)$. Further, use π_{ij} to denote the transition probability $P(i \rightarrow j)$, that is, $\pi_{ij} = P(i \rightarrow j)$. Finally, use p_Y to denote $P(Y)$, p_Y^L to denote $P(L|Y)$, $p_{\overline{Y}}^L$ to denote $P(L|\overline{Y})$, and so on. As an example, Equation 2.24 above then becomes

$$\pi_{11} = 1 - (p_Y^L p_Y + p_{\overline{Y}}^{\overline{L}} p_{\overline{Y}}). \quad (2.25)$$

Filling out the $\mathbf{M}$-matrix, you get:

$$\begin{bmatrix} 1-(p_Y^L p_Y + p_{\overline{Y}}^{\overline{L}} p_{\overline{Y}}) & p_Y^L p_Y + p_{\overline{Y}}^{\overline{L}} p_{\overline{Y}} & 0 & 0 \\ \frac{1}{s} p_Y^{\overline{L}} p_Y & 1-(\frac{1}{s} p_Y^{\overline{L}} p_Y + p_Y^L p_Y + p_{\overline{Y}}^{\overline{L}} p_Y) & p_Y^L p_Y + p_{\overline{Y}}^{\overline{L}} p_Y & 0 \\ 0 & \frac{1}{s} p_Y^{\overline{L}} p_Y & 1-(\frac{1}{s} p_Y^{\overline{L}} p_Y + p_Y^L p_Y) & p_Y^L p_Y \\ 0 & 0 & \frac{1}{s} p_Y^{\overline{L}} p_Y & 1-\frac{1}{s} p_Y^{\overline{L}} p_Y \end{bmatrix} \quad (2.26)$$

Referring to Figure 2.8, notice how row i covers all the outgoing transitions of State i. As always, since you either have to remain in the current state or go to one of the other states, you get

$$\sum_{j=1}^{4} \pi_{ij} = 1 \tag{2.27}$$

for each row i. Correspondingly, column j covers all the incoming transitions of State j.

Numeric Single-Literal Learning Outcome

The $\mathbf{M}$ matrix provides the transition probabilities of a single step of learning. Now, collect your state probabilities in a vector $\boldsymbol{\pi}$:

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 \end{bmatrix}. \tag{2.28}$$

Then you get the state probabilities after a single learning step by multiplying $\boldsymbol{\pi}$ with $\mathbf{M}$:

$$\boldsymbol{\pi}\mathbf{M}. \tag{2.29}$$

Further, you get the state probabilities after two steps of learning by multiplying with $\mathbf{M}$ squared:

$$\boldsymbol{\pi}\mathbf{M}^2. \tag{2.30}$$

By performing a sufficiently large number of learning steps, e.g., 10000 steps, you approximate the stationary distribution of single-literal learning:

$$\boldsymbol{\pi}\mathbf{M}^{10000}. \tag{2.31}$$

Initialization. To perform the above calculations in practice, all the entries in $\boldsymbol{\pi}$ and $\mathbf{M}$ must be replaced with actual probability values. Typically, you initialize a four-state Literal Automaton to be in State 2:

$$\boldsymbol{\pi} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}, \tag{2.32}$$

or, alternatively, randomly in either State 2 or State 3:

$$\boldsymbol{\pi} = \begin{bmatrix} 0 & 0.5 & 0.5 & 0 \end{bmatrix}. \tag{2.33}$$

Remark. Note that the stationary distribution of a Markov chain does not depend on the initial state probabilities $\boldsymbol{\pi}$, so you can initialize the state probabilities any way you like.

Remark. The probability values in the transition matrix $\mathbf{M}$ depends on the classification task at hand. You will study several different classification tasks later in this chapter.

Analytic Single-Literal Learning Outcome (Optional)

If you like, you can now jump ahead to the next subsection to study the stationary distribution of single-literal learning. Otherwise, if you prefer to calculate the stationary distribution exactly, proceed as follows.

In the same way you found the stationary probabilities of *Rainy* and *Sunny* weather, you find the stationary state probabilities $\boldsymbol{\pi}$. Again, you seek the state probabilities that remain unaffected by the learning step. Using matrix form makes the formulation compact:

$$\boldsymbol{\pi} = \boldsymbol{\pi}\mathbf{M}. \tag{2.34}$$

Writing out each equation, you get:

$$\pi_1 = \pi_1 \left(1 - p_Y^L p_Y - p_{\overline{Y}}^{\overline{L}} p_{\overline{Y}}\right) + \pi_2 \frac{p_Y^{\overline{L}} p_Y}{s} \tag{2.35}$$

$$\pi_2 = \begin{array}{l} \pi_1 \left(p_Y^L p_Y p_Y + p_{\overline{Y}}^{\overline{L}} p_{\overline{Y}}\right) + \\ \pi_2 \left(1 - \frac{p_Y^{\overline{L}} p_Y}{s} - p_Y^L p_Y - p_{\overline{Y}}^{\overline{L}} p_{\overline{Y}}\right) + \pi_3 \frac{p_Y^{\overline{L}} p_Y}{s} \end{array} \tag{2.36}$$

$$\pi_3 = \pi_2 \left(p_Y^L p_Y + p_{\overline{Y}}^{\overline{L}} p_{\overline{Y}}\right) + \pi_3 \left(1 - \frac{p_Y^{\overline{L}} p_Y}{s} - p_Y^L p_Y\right) + \pi_4 \frac{p_Y^{\overline{L}} p_Y}{s} \tag{2.37}$$

$$\pi_4 = \pi_3 p_Y^L p_Y + \pi_4 \left(1 - \frac{p_Y^{\overline{L}} p_Y}{s}\right). \tag{2.38}$$

For instance, Equation 2.35 says that the state probabilities π_1 and π_2 are set so that π_1 does not change, despite either staying in place ($1 \rightarrow 1$) or moving from State 2 to State 1 ($2 \rightarrow 1$).

Finally, to solve the equation system, you also need to add the constraint $\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$.

		1.	2.	3.	4.	5.	6.
π_1	=	α		$P(\overline{L}\|Y)^3$			$P(Y)^2$
π_2	=	α		$P(\overline{L}\|Y)^2$	s	$\big(P(L\|Y)P(Y) + P(\overline{L}\|\overline{Y})P(\overline{Y})\big)$	$P(Y)$
π_3	=	α		$P(\overline{L}\|Y)$	s^2	$\big(P(L\|Y)P(Y) + P(\overline{L}\|\overline{Y})P(\overline{Y})\big)^2$	
π_4	=	α	$P(L\|Y)$		s^3	$\big(P(L\|Y)P(Y) + P(\overline{L}\|\overline{Y})P(\overline{Y})\big)^2$	

Table 2.1: Stationary distribution of four-state Literal Automaton.

Analysis of Single-Literal Learning

Table 2.1 contains the stationary distribution of the four-state Literal Automaton (found by solving Equation 2.34). Each state probability π_1, π_2, π_3, and π_4 gets its own expression — a multiplication of several distinct factors.

Analysis of Expressions. To make it more easy to analyse the expressions, the table organizes the factors in columns:

1. The first column contains the normalization factor α, which ensures that the state probabilities sum to 1: $\pi_1 + \pi_2 + \pi_3 + \pi_4 = 1$.

2. The second column consists of $P(L|Y)$. A high $P(L|Y)$ means that the literal is frequently *True* for Y. Observe how an increasing $P(L|Y)$ shifts the probability distribution to State 4 (Memorized). *The Tsetlin machine learns frequent patterns.*

3. The third column covers $P(\overline{L}|Y)$. A high $P(\overline{L}|Y)$ means that the literal is infrequently *True* for Y because $P(L|Y) = 1 - P(\overline{L}|Y)$. Notice how the probability distribution shifts towards State 1 and State 2 (Forgotten) with increasing $P(\overline{L}|Y)$. *The Tsetlin machine forgets infrequent patterns.*

4. The fourth column contains the symbol s from the Forget Value $\frac{1}{s}$. A high s counters the effect of a high $P(\overline{L}|Y)$ in column two, shifting the probability distribution back towards State 4 (Memorized). *The Forget Value $\frac{1}{s}$ allows you to control the frequency of the patterns learnt.*

5. The fifth column introduces: $P(L|Y)P(Y) + P(\overline{L}|\overline{Y})P(\overline{Y})$. A high $P(L|Y)P(Y)$ in combination with a high $P(\overline{L}|\overline{Y})P(\overline{Y})$ means that the

literal both recognizes Y and rules out $\overline{Y}$, making it discriminative. *The Tsetlin machine seeks discriminative patterns.*

6. The sixth column covers $P(Y)$, which balances the other factors. A higher $P(Y)$ increases the probability of staying in State 1 and State 2 (Forgotten).

Finally, observe how each factor increases/decreases exponentially towards Maximally Forgotten/Maximally Memorized. *Increasing the number of Literal Automaton states improves learning accuracy exponentially.*

Analysis of Example Classification Scenarios. The following four classification scenarios provide further insight on the learning outcome.

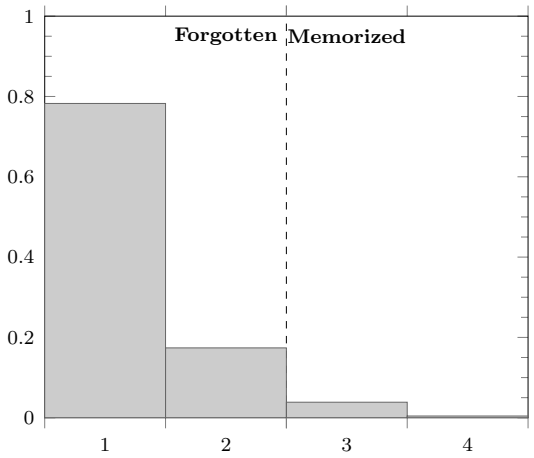

Figure 2.9: Stationary distribution of four-state Literal Automaton when $s = 1.0$, $P(Y) = 0.9$, $P(L|Y) = 0.1$, $P(\overline{Y}) = 0.1$, and $P(\overline{L}|\overline{Y}) = 0.9$. The probability of *memorizing* the literal is $\pi_3 + \pi_4 = 0.04$.

- **Scenario 1: Tsetlin Machines Forget Infrequent Patterns.** The first scenario involves a low frequency pattern and quick forgetting with $s = 1.0$. That is, the class Y is frequent because $P(Y) = 0.9$. However, the literal is *True* for Y only with probability 0.1,

$P(L|Y) = 0.1$, making it infrequent. This means that the (L, Y)-observation occurs with probability $P(L|Y)P(Y) = 0.1 \cdot 0.9 = 0.09$. Additionally, class $\overline{Y}$ is infrequent with $P(\overline{Y}) = 0.1$. The probability of the literal being *False* for $\overline{Y}$ is high, on the other hand: $P(\overline{L}|\overline{Y}) = 0.9$. Employing Bayes rule you get: $P(Y|L) = \frac{P(L|Y)P(Y)}{P(L)} = \frac{P(L|Y)P(Y)}{P(L|Y)P(Y)+(1-P(\overline{L}|\overline{Y}))P(\overline{Y})} = \frac{0.09}{0.09+0.01} = 0.9 = P(Y)$. In other words, Y and L are independent. The independence means that the literal does not have any discrimination power. Figure 2.9 shows the learning outcome when using a four-state Literal Automaton. Notice that the Tsetlin machine memorizes the literal with low probability: $\pi_3 + \pi_4 = 0.04$.

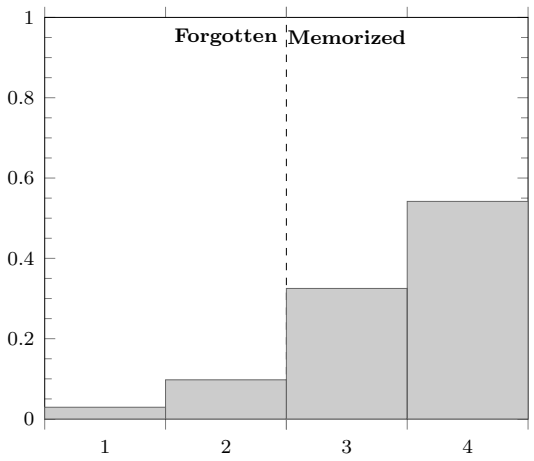

Figure 2.10: Stationary distribution of four-state Literal Automaton when $s = 15.0$, $P(Y) = 0.9$, $P(L|Y) = 0.1$, $P(\overline{Y}) = 0.1$, and $P(\overline{L}|\overline{Y}) = 0.9$. The probability of *memorizing* the literal is $\pi_3 + \pi_4 = 0.87$.

- **Scenario 2: Slower Forgetting Retains Less Frequent Patterns.** Continue with the above scenario, but increase s to 15.0. This means that the Tsetlin machine looks for patterns with frequency above $\frac{1}{15.0} \approx 0.07$. Figure 2.10 shows the new learning outcome. Observe how the Tsetlin machine with $s = 15.0$ now memorizes the literal with probability $\pi_3 + \pi_4 = 0.87$.

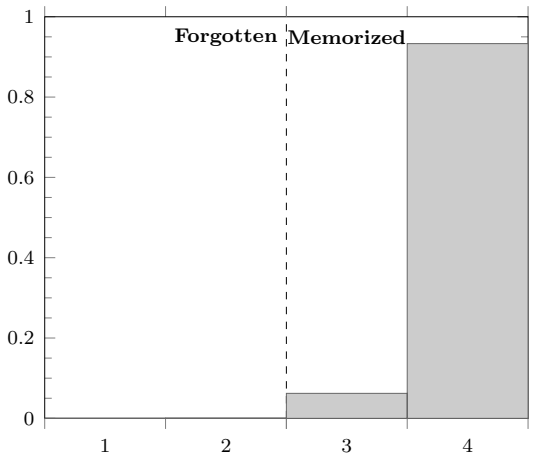

Figure 2.11: Stationary distribution of four-state Literal Automaton when $s = 15.0$, $P(Y) = 0.82$, $P(L|Y) = 0.99$, $P(\overline{Y}) = 0.18$, and $P(\overline{L}|\overline{Y}) = 0.5$. The probability of *memorizing* the literal is $\pi_3 + \pi_4 = 0.9999$.

- **Scenario 3: Tsetlin Machines Prioritize Patterns That Discriminate Between Classes.** In this scenario, the literal predicts class Y accurately, ruling out class $\overline{Y}$. That is, $P(Y) = 0.82$, $P(L|Y) = 0.99$, $P(\overline{Y}) = 0.18$, and $P(\overline{L}|\overline{Y}) = 0.5$. Using Bayes rule you get $P(Y|L) = \frac{P(L|Y)P(Y)}{P(L)} = \frac{0.8118}{0.9018} = 0.90 \neq P(Y)$. Accordingly, the literal is now discriminative. Figure 2.11 shows the learning outcome when using a four-state Literal Automaton. Notice that the Tsetlin machine includes the literal in the rule with high probability: $\pi_3 + \pi_4 = 0.9999$.
- **Scenario 4: Deeper Memory Increases Learning Accuracy.** Proceed from Scenario 3 by increasing the memory depth to eight. Figure 2.12 shows how the deeper memory makes the stationary distribution move away from the center. As a result, memorizing the literal becomes almost certain: $\pi_5 + \pi_6 + \pi_7 + \pi_8 \approx 1.0$.

2.2 Multi-Literal Learning

You are now ready to investigate how the Tsetlin machine coordinates the learning of multiple literals to build a rule. The tool to use is a *two-player*

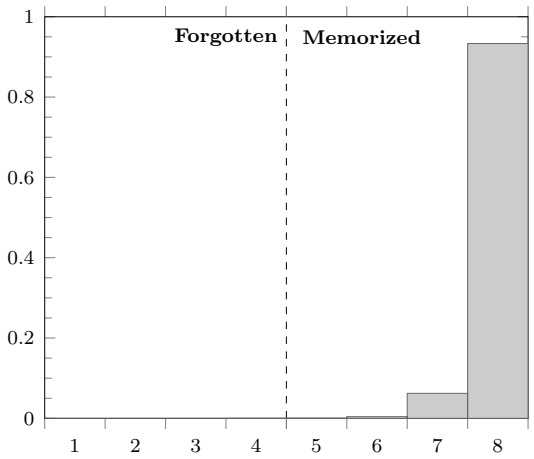

Figure 2.12: Stationary distribution of eight-state Literal Automaton when $s = 15.0$, $P(Y) = 0.82$, $P(L|Y) = 0.99$, $P(\overline{Y}) = 0.18$, and $P(\overline{L}|\overline{Y}) = 0.5$. The probability of *memorizing* the literal is $\pi_5 + \pi_6 + \pi_7 + \pi_8 \approx 1.0$.

game from *game theory* — a methodology for studying how decision-makers interact.

Prisoner's Dilemma

The well-known Prisoner's Dilemma game showcases the essence of two-player games. In Prisoner's Dilemma, the police have imprisoned two crime partners in separate cells. The partners cannot communicate with each other and get the following choice. They can *Betray* their partner, or they can *Remain Silent*. The dilemma is as follows. If both *Remain Silent*, the police have little evidence, and the partners get one year in prison each. However, if one *Betrays* the other and the other *Remains Silent*, the police release the traitor. The unfortunate betrayed one gets twenty years in prison. This situation leads each partner to *Betray* the other, but in that case, they end up with five years in prison each. They would have been better off with *Remain Silent*!

Game Matrix. The matrix in Table 2.2 describes the game more formally. It represents the interaction between the two prisoners and is called

		Prisoner Two	
		Remain Silent	Betray
Prisoner One	Remain Silent	*One Year* / *One Year*	*Released* / *Twenty Years*
	Betray	*Twenty Years* / *Released*	*Five Years* / *Five Years*

Table 2.2: A two-player game – Prisoner's Dilemma.

a *game matrix*. *Prisoner One* selects a matrix row, and *Prisoner Two* selects a matrix column. The combined choice singles out a matrix cell that defines the choice's outcome. The lower left punishment in the cell is for *Prisoner One*. The upper right is for *Prisoner Two*. For example, if *Prisoner One* selects *Remain Silent* and *Prisoner Two* selects *Betray*, the upper right cell contains their punishments. *Prisoner One* gets *Twenty Years* in prison, while *Prisoner Two* is *Released*.

Figure 2.13: Memory of depth eight with two literals: *Literal One* and *Literal Two*. Being in memory position 5, they are both *Memorized*. The resulting rule is: **if** *Literal One* **and** *Literal Two* **then** Y.

Literal Game

Literals in Tsetlin machine learning face a similar situation as the prisoners in Prisoner's Dilemma. See, for example, the rule memory in Figure 2.13. It contains the two literals *Literal One* and *Literal Two*. Without communicating with each other, these decide to either be *Memorized* or *Forgotten*. That is, they choose between taking part in the rule or not taking part. By analyzing a game of literals, you will understand how they can make this decision successfully. No communication explains why the Tsetlin machine learning algorithm is so short — coordination emerges from the Recognize, Erase, and Reject Feedback.

Rule. Depending on the decisions of the literals you get one of the following rules:

1. **if** *True* **then** Y.
2. **if** *Literal One* **then** Y.
3. **if** *Literal Two* **then** Y.
4. **if** *Literal One* **and** *Literal Two* **then** Y.

Note that the condition of the first rule is empty. During learning, an empty condition is *True* by default. Of course, which rule you get depends on the observations you use to update the memory. Recall that a Tsetlin machine builds many rules, also for class $\overline{Y}$. You here study the construction of a single rule for class Y only.

Observations. A Literal Game fuses many different games, one game per kind of observation. Remember how a single literal gives four sorts of observations: (L, Y), $(\overline{L}, Y)$, $(L, \overline{Y})$, and $(\overline{L}, \overline{Y})$. With an extra literal, the number doubles to eight: (L_1, L_2, Y), $(\overline{L}_1, L_2, Y)$, $(L_1, \overline{L}_2, Y)$, $(\overline{L}_1, \overline{L}_2, Y)$, $(L_1, L_2, \overline{Y})$, $(\overline{L}_1, L_2, \overline{Y})$, $(L_1, \overline{L}_2, \overline{Y})$, and $(\overline{L}_1, \overline{L}_2, \overline{Y})$. Here, L_1 means that *Literal One* is *True*, while $\overline{L}_1$ means that *Literal One* is *False*, and so on. The resulting eight games reveal the outcome of Tsetlin machine learning.

Literal One \ Literal Two	Forgotten	Memorized
Forgotten	$\emptyset$ *True*	L_2 *True*
Memorized	$\overline{L}_1$ *False*	$\overline{L}_1 \wedge L_2$ *False*

Literal One \ Literal Two	Forgotten	Memorized
Forgotten	$\Uparrow$ $\downarrow$	$\Uparrow$ $\downarrow$
Memorized	$\downarrow$ $\downarrow$	$\downarrow$ $\downarrow$

Table 2.3: Literal Game for observation $(\overline{L}_1, L_2, Y)$.

Example Game. An example with observation $(\overline{L}_1, L_2, Y)$ explains how to create the eight games. First, take a look at the left matrix in Table 2.3. *Literal One* is either *Forgotten* or *Memorized*. The same goes for *Literal Two*. Notice how each matrix cell shows which literals take part in the rule's condition. The cell also contains the condition's truth value for the $(\overline{L}_1, L_2, Y)$-observation. This is all you need to determine the outcomes of the $(\overline{L}_1, L_2, Y)$-game. The right matrix in Table 2.3 shows these outcomes. Let us explore each cell of the matrix:

1. The (*Literal One Forgotten*, *Literal Two Forgotten*) cell gives a rule without literals — the rule's condition is *empty*. Use the symbol $\emptyset$ for empty conditions. Recall that an empty condition is *True* by definition. Further, the observed class $(\overline{L}_1, L_2, Y)$ is the same as the rule's class. As you know from Chapter 1, you then get Recognize Feedback. Recognize Feedback pushes *False* literals downwards and *True* literals upwards in memory. The downward arrow $\downarrow$ in the cell means push *Literal One* downwards (forget it). The upward arrow $\Uparrow$ says pull *Literal Two* upwards (memorize it). Figure 2.14 illustrates such an update. Note that the upward arrow is double to signify that memorization typically is stronger than forgetting (e.g., with Memorize Value 1.0 and Forget Value $\frac{1}{s}$).

2. The (*Literal One Memorized*, *Literal Two Forgotten*) cell produces a rule with *Literal One* only. *Literal One* is *False* in observation $(\overline{L}_1, L_2, Y)$ so the rule's condition is *False*. Since the observed class is the same as the rule's class, you get Erase Feedback. The two

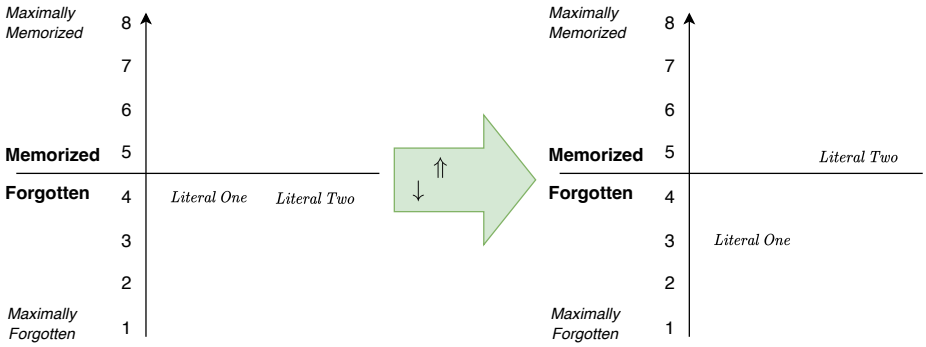

Figure 2.14: Memory update for observation $(\overline{L}_1, L_2, Y)$, giving Recognize Feedback. Based on their truth values, the feedback pushes *Literal One* downwards and pulls *Literal Two* upwards in memory.

downward arrows $\downarrow\downarrow$ mean: push both literals downwards in memory (forget them).

3. The (*Literal One Forgotten*, *Literal Two Memorized*) cell yields a rule with *Literal Two*. Since *Literal Two* in $(\overline{L}_1, L_2, Y)$ is *True*, the rule's condition is *True*. Again, you get Recognize Feedback.

4. Finally, the (*Literal One Memorized*, *Literal Two Memorized*) cell gives a rule with both literals. Let $\wedge$ be shorthand notation for **and**. The rule's condition $\overline{L}_1 \wedge L_2$ is *False* because *Literal One* is *False*. Yet again, you get Erase Feedback.

Complete List of Games. Using Recognize, Erase, and Reject Feedback from Chapter 1, the above approach applies to all of the eight kinds of observations. The matrix in Table 2.4 specifies the resulting games. Each matrix cell lists the possible observations in the first column (Obs.). The second column (Rule) calculates the truth value of the rule. Finally, the third column lists the game outcome. The outcome is either memorize literal ($\Uparrow$), forget literal ($\downarrow$), or nothing ($-$). The first outcome in the pairs applies to *Literal One* and the second to *Literal Two*.

	Literal Two Forgotten			Literal Two Memorized		
Literal One Forgotten	**Obs.**	**Rule**	**Outcome**	**Obs.**	**Rule**	**Outcome**
	(L_1, L_2, Y)	$\emptyset = \mathit{True}$	$\Uparrow\Uparrow$	(L_1, L_2, Y)	$L_2 = \mathit{True}$	$\Uparrow\Uparrow$
	$(L_1, \overline{L}_2, Y)$	$\emptyset = \mathit{True}$	$\Uparrow\downarrow$	$(L_1, \overline{L}_2, Y)$	$\overline{L}_2 = \mathit{False}$	$\downarrow\downarrow$
	$(\overline{L}_1, L_2, Y)$	$\emptyset = \mathit{True}$	$\downarrow\Uparrow$	$(\overline{L}_1, L_2, Y)$	$L_2 = \mathit{True}$	$\downarrow\Uparrow$
	$(\overline{L}_1, \overline{L}_2, Y)$	$\emptyset = \mathit{True}$	$\downarrow\downarrow$	$(\overline{L}_1, \overline{L}_2, Y)$	$\overline{L}_2 = \mathit{False}$	$\downarrow\downarrow$
	$(L_1, L_2, \overline{Y})$	$\emptyset = \mathit{True}$	$--$	$(L_1, L_2, \overline{Y})$	$L_2 = \mathit{True}$	$--$
	$(L_1, \overline{L}_2, \overline{Y})$	$\emptyset = \mathit{True}$	$-\Uparrow$	$(L_1, \overline{L}_2, \overline{Y})$	$\overline{L}_2 = \mathit{False}$	$--$
	$(\overline{L}_1, L_2, \overline{Y})$	$\emptyset = \mathit{True}$	$\Uparrow -$	$(\overline{L}_1, L_2, \overline{Y})$	$L_2 = \mathit{True}$	$\Uparrow -$
	$(\overline{L}_1, \overline{L}_2, \overline{Y})$	$\emptyset = \mathit{True}$	$\Uparrow\Uparrow$	$(\overline{L}_1, \overline{L}_2, \overline{Y})$	$\overline{L}_2 = \mathit{False}$	$--$
Literal One Memorized	**Obs.**	**Rule**	**Outcome**	**Obs.**	**Rule**	**Outcome**
	(L_1, L_2, Y)	$L_1 = \mathit{True}$	$\Uparrow\Uparrow$	(L_1, L_2, Y)	$L_1 \wedge L_2 = \mathit{True}$	$\Uparrow\Uparrow$
	$(L_1, \overline{L}_2, Y)$	$L_1 = \mathit{True}$	$\Uparrow\downarrow$	$(L_1, \overline{L}_2, Y)$	$L_1 \wedge \overline{L}_2 = \mathit{False}$	$\downarrow\downarrow$
	$(\overline{L}_1, L_2, Y)$	$\overline{L}_1 = \mathit{False}$	$\downarrow\downarrow$	$(\overline{L}_1, L_2, Y)$	$\overline{L}_1 \wedge L_2 = \mathit{False}$	$\downarrow\downarrow$
	$(\overline{L}_1, \overline{L}_2, Y)$	$\overline{L}_1 = \mathit{False}$	$\downarrow\downarrow$	$(\overline{L}_1, \overline{L}_2, Y)$	$\overline{L}_1 \wedge \overline{L}_2 = \mathit{False}$	$\downarrow\downarrow$
	$(L_1, L_2, \overline{Y})$	$L_1 = \mathit{True}$	$--$	$(L_1, L_2, \overline{Y})$	$L_1 \wedge L_2 = \mathit{True}$	$--$
	$(L_1, \overline{L}_2, \overline{Y})$	$L_1 = \mathit{True}$	$-\Uparrow$	$(L_1, \overline{L}_2, \overline{Y})$	$L_1 \wedge \overline{L}_2 = \mathit{False}$	$--$
	$(\overline{L}_1, L_2, \overline{Y})$	$\overline{L}_1 = \mathit{False}$	$--$	$(\overline{L}_1, L_2, \overline{Y})$	$\overline{L}_1 \wedge L_2 = \mathit{False}$	$--$
	$(\overline{L}_1, \overline{L}_2, \overline{Y})$	$\overline{L}_1 = \mathit{False}$	$--$	$(\overline{L}_1, \overline{L}_2, \overline{Y})$	$\overline{L}_1 \wedge \overline{L}_2 = \mathit{False}$	$--$

Table 2.4: The eight Literal Games in one matrix.

Markov Chain of Literal Game

You are now about to discover the outcome of the two-literal Tsetlin machine learning process. The final piece of information you need is the observation probabilities $P(L_1, L_2|Y)P(Y)$, $P(\overline{L}_1, L_2|Y)P(Y)$, $P(L_1, \overline{L}_2|Y)P(Y)$, $P(\overline{L}_1, \overline{L}_2|Y)P(Y)$, $P(L_1, L_2|\overline{Y})P(\overline{Y})$, $P(\overline{L}_1, L_2|\overline{Y})P(\overline{Y})$, $P(L_1, \overline{L}_2|\overline{Y})P(\overline{Y})$, and $P(\overline{L}_1, \overline{L}_2|\overline{Y})P(\overline{Y})$. These probabilities define the learning task at hand and vary from task to task.

Transition Graph of Literal Game. You follow the same procedure as you did for the single-literal case. First, you construct the Markov chain of the learning process from the observation probabilities and the Recognize, Erase, and Reject Feedback. Using a single literal with four memory positions, you got a four state Markov chain. With two literals, however, the Markov chain gets $4 \times 4 = 16$ states. Figure 2.15 shows the reason. You get one Markov chain state per combination of literal memory positions. In

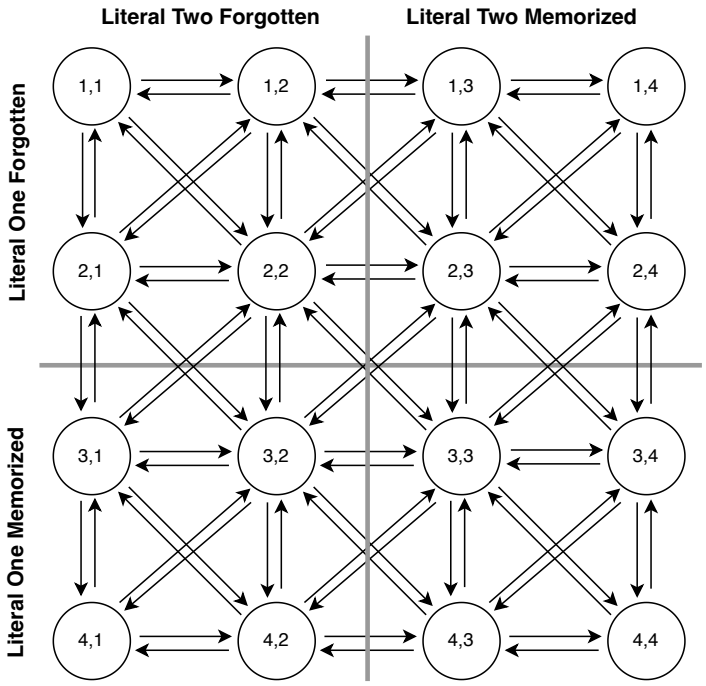

Figure 2.15: States and transitions of the Markov chain for a literal pair, each with four memory positions.

each Markov chain state in the figure, the first number is the memory position of *Literal One*. The second number is the memory position of *Literal Two*. Observe how both of the literals can change position in memory in the same Markov chain transition. This means that when you for instance are in state $(3, 2)$, you can transition to nine different states: $(2, 1)$, $(2, 2)$, $(2, 3)$, $(3, 1)$, $(3, 2)$, $(3, 3)$, $(4, 1)$, $(4, 2)$, and $(4, 3)$.

To finish building the Markov chain, you need the probability of each of these transitions. Three example transition probabilities explain the procedure to follow. Figure 2.16 shows the states and transitions for memory depth two. With only two memory positions, the Markov chain states correspond directly to the Literal Game cells in Table 2.4. The state FF in the Markov chain, for instance, corresponds to the (*Literal One Forgotten*, *Literal Two Forgotten*)-cell of the Literal Game.

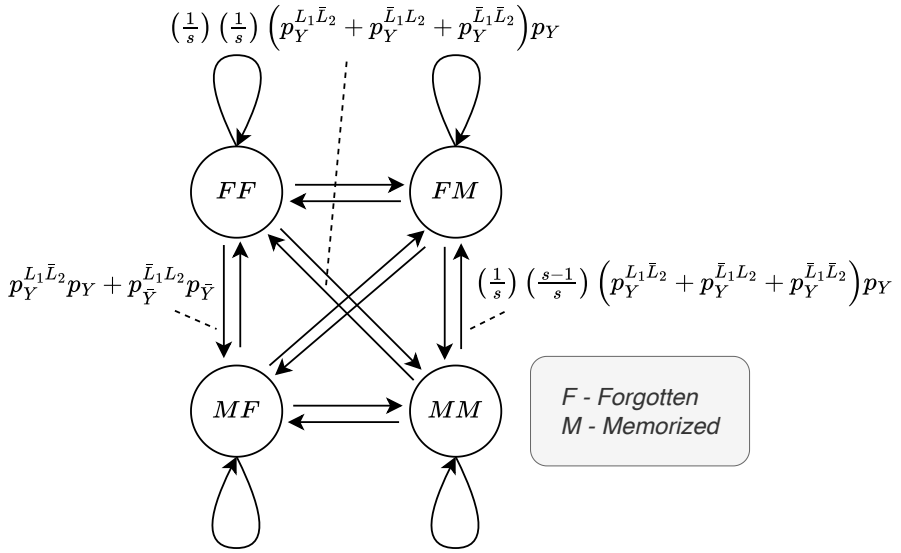

Figure 2.16: Transition graph of Markov chain for two interacting literals.

Transition $MM \rightarrow FM$. The starting state of this transition is MM. This state means that we are in the lower right quadrant of Table 2.4. The quadrant reveals that only observations $(L_1, \overline{L}_2, Y)$, $(\overline{L}_1, L_2, Y)$, and $(\overline{L}_1, \overline{L}_2, Y)$ give the outcome $\downarrow\downarrow$, which makes the transition possible. Additionally, to get to the FM-state, *Literal One* must change its memory position. This happens with probability $\frac{1}{s}$. Further, *Literal Two* must stay in place, which happens with probability $\frac{s-1}{s}$. Accordingly, you get the following transition probability:

$$
\begin{aligned}
P(MM \rightarrow FM) &= \left(\tfrac{1}{s}\right)\left(\tfrac{s-1}{s}\right) p_Y^{L_1\overline{L}_2} p_Y + && (2.39)\\
&\quad \left(\tfrac{1}{s}\right)\left(\tfrac{s-1}{s}\right) p_Y^{\overline{L}_1 L_2} p_Y + && (2.40)\\
&\quad \left(\tfrac{1}{s}\right)\left(\tfrac{s-1}{s}\right) p_Y^{\overline{L}_1\overline{L}_2} p_Y && (2.41)\\
&= \left(\tfrac{1}{s}\right)\left(\tfrac{s-1}{s}\right)\left(p_Y^{L_1\overline{L}_2} + p_Y^{\overline{L}_1 L_2} + p_Y^{\overline{L}_1\overline{L}_2}\right) p_Y. && (2.42)
\end{aligned}
$$

Transition $MM \rightarrow FF$. With the same starting state as above, you are still in the lower right quadrant of Table 2.4. However, now both of the

literals need to change position in memory to get to state FF:

$$P(MM \rightarrow FF) \;=\; \left(\tfrac{1}{s}\right)\left(\tfrac{1}{s}\right)\left(p_Y^{L_1\overline{L}_2} + p_Y^{\overline{L}_1 L_2} + p_Y^{\overline{L}_1\overline{L}_2}\right) p_Y. \quad (2.43)$$

Transition $FF \rightarrow MF$. This transition starts in state FF. The state corresponds to the upper left quadrant of Table 2.4. Observation $(L_1, \overline{L}_2, Y)$ yields the outcome $\Uparrow\downarrow$. The down arrow $\downarrow$ becomes ineffective because the literal is already Maximally Forgotten. Finally, observation $(\overline{L}_1, L_2, \overline{Y})$ gives the outcome $\Uparrow -$. Accordingly, the probability of transitioning to MF becomes:

$$P(FF \rightarrow MF) \;=\; p_Y^{L_1\overline{L}_2} p_Y + p_{\overline{Y}}^{\overline{L}_1 L_2} p_{\overline{Y}}. \quad (2.44)$$

Using Table 2.4 in the above manner, you can construct the transition matrix $\mathbf{M}$, which collects all the transition probabilities. With an initial state distribution $\boldsymbol{\pi}$ and the transition matrix $\mathbf{M}$ in place, you proceed as for the single-literal case. You find the outcome of two-literal Tsetlin machine learning in the stationary distribution of the Markov chain:

$$\boldsymbol{\pi}\mathbf{M}^{10000}. \quad (2.45)$$

Using the stationary distribution, four scenarios give insight into how a Tsetlin machine coordinates literal learning.

Analysis of Two-Literal Learning

Scenario 1: Tsetlin Machines Select Informative Literals. In the first scenario, the patterns are equally frequent: $P(L_1 L_2) = P(\overline{L}_1 L_2) = P(L_1\overline{L}_2) = P(\overline{L}_1\overline{L}_2) = 0.25$. Only *Literal Two* provides information on the class. If *Literal Two* is *True*, you predict the class with precision 0.9: $P(Y|L_1 L_2) = 0.9$ and $P(Y|\overline{L}_1 L_2) = 0.9$. If only *Literal One* is *True*, however, you do not get any information on the class: $P(Y|L_1\overline{L}_2) = 0.5$. Finally, when both literals are *False*, class $\overline{Y}$ is most likely: $P(Y|\overline{L}_1\overline{L}_2) = 0.2$. Table 2.5 contains the learning outcome for eight memory positions and $s = 1.0$. Notice how the Tsetlin machine picks out the informative literal – *Literal Two*. Also, observe how the probability distribution peaks in the upper right quadrant where *Literal One* is in memory position 1 and *Literal Two* is in memory position 8.

		Literal Two Forgotten				Literal Two Memorized			
		1	2	3	4	5	6	7	8
Lit. One Forg.	1	.00	.00	.00	.01	.02	.06	.11	.16
	2	.00	.00	.00	.00	.01	.02	.05	.14
	3	.00	.00	.00	.00	.01	.01	.03	.10
	4	.00	.00	.00	.01	.01	.01	.02	.05
Lit. One Mem.	5	.00	.00	.00	.00	.01	.01	.01	.03
	6	.00	.00	.00	.00	.00	.01	.01	.01
	7	.00	.00	.00	.00	.00	.00	.01	.01
	8	.00	.00	.00	.00	.00	.00	.00	.01

Table 2.5: Learning outcome with two literal automata when $s = 1.0$, $P(L_1L_2) = P(\overline{L}_1L_2) = P(L_1\overline{L}_2) = P(\overline{L}_1\overline{L}_2) = 0.25$, $P(Y|L_1L_2) = 0.9$, $P(Y|\overline{L}_1L_2) = 0.9$, $P(Y|L_1\overline{L}_2) = 0.5$, and $P(Y|\overline{L}_1\overline{L}_2) = 0.2$. A double line marks the decision boundary: *Forgotten* vs. *Memorized*.

		Literal Two Forgotten				Literal Two Memorized			
		1	2	3	4	5	6	7	8
Lit. One Forg.	1	.00	.00	.00	.01	.02	.04	.06	.07
	2	.00	.00	.00	.00	.01	.02	.03	.06
	3	.00	.00	.00	.00	.01	.01	.02	.04
	4	.01	.00	.00	.01	.01	.01	.01	.02
Lit. One Mem.	5	.02	.01	.01	.01	.01	.00	.01	.01
	6	.04	.02	.01	.01	.00	.01	.00	.01
	7	.06	.03	.02	.01	.01	.00	.01	.00
	8	.07	.06	.04	.02	.01	.01	.00	.01

Table 2.6: Learning outcome with two literals when $s = 1.0$, $P(L_1L_2) = P(\overline{L}_1L_2) = P(L_1\overline{L}_2) = P(\overline{L}_1\overline{L}_2) = 0.25$, $P(Y|L_1L_2) = 0.99$, $P(Y|\overline{L}_1L_2) = 0.9$, $P(Y|L_1\overline{L}_2) = 0.9$, and $P(Y|\overline{L}_1\overline{L}_2) = 0.1$. A double line marks the decision boundary: *Forgotten* vs. *Memorized*.

Scenario 2: Tsetlin Machines Learn Multiple Alternatives. In the second scenario, both literals are informative. That is, when only one of them is *True*, your precision becomes 0.9: $P(Y|\overline{L}_1L_2) = 0.9$ and $P(Y|L_1\overline{L}_2) = 0.9$. When both are true you get precision 0.99: $P(Y|L_1L_2) = 0.99$. Again, the observations occur with equal probability: $P(L_1L_2) =$

$P(\overline{L}_1 L_2) = P(L_1 \overline{L}_2) = P(\overline{L}_1 \overline{L}_2) = 0.25$. With an s-value of 1.0, only a single literal gives sufficiently high pattern frequency. You find the learning outcome in Table 2.6. Observe the two peaks, one where *Literal One* takes part in the rule and one where *Literal Two* takes part. Throughout a single learning session, you will observe each alternative, and the Tsetlin machine will continue to revisit both as learning continues.

		Literal Two Forgotten				Literal Two Memorized			
		1	2	3	4	5	6	7	8
Lit. One Forg.	1	.00	.00	.00	.01	.02	.05	.09	.12
	2	.00	.00	.00	.00	.01	.02	.05	.11
	3	.00	.00	.00	.00	.01	.01	.02	.08
	4	.00	.00	.00	.01	.01	.01	.02	.04
Lit. One Mem.	5	.01	.01	.01	.01	.02	.01	.01	.02
	6	.01	.01	.01	.01	.00	.01	.01	.01
	7	.01	.01	.01	.01	.01	.00	.01	.01
	8	.01	.01	.01	.01	.01	.00	.00	.01

Table 2.7: Learning outcome with two literals when $s = 1.0$, $P(L_1 L_2) = P(\overline{L}_1 L_2) = P(L_1 \overline{L}_2) = P(\overline{L}_1 \overline{L}_2) = 0.25$, $P(Y|L_1 L_2) = 0.97$, $P(Y|L_1 \overline{L}_2) = 0.7$, $P(Y|\overline{L}_1 L_2) = 0.9$, and $P(Y|\overline{L}_1 \overline{L}_2) = 0.1$. A double line marks the decision boundary: *Forgotten* vs. *Memorized*.

Scenario 3: Tsetlin Machines Pick the Best Literal. In the third scenario, we reduce the prediction power of *Literal One* to $P(Y|L_1 \overline{L}_2) = 0.7$. It is still informative, yet, now *Literal Two* is the better choice. Table 2.7 shows the learning outcome. Note how the learning outcome distribution now peaks at memory position 1 for *Literal One* and position 8 for *Literal Two*. In other words, the learning outcome singles out the most informative literal.

Scenario 4: Tsetlin Machines Increase Precision by Combining Literals. When you increase s in Scenario 2 from 1.0 to 4.0, you allow the Tsetlin machine to learn less frequent patterns. The learning outcome is in Table 2.8. Notice how the increase in s shifts the peak of the stationary distribution to memory position 8 for both literals. This produces a rule

		Literal Two Forgotten				Literal Two Memorized			
		1	2	3	4	5	6	7	8
Lit. One Forg.	1	.00	.00	.00	.00	.00	.00	.00	.00
	2	.00	.00	.00	.00	.00	.00	.00	.00
	3	.00	.00	.00	.00	.00	.00	.00	.01
	4	.00	.00	.00	.00	.00	.00	.01	.01
Lit. One Mem.	5	.00	.00	.00	.00	.01	.01	.02	.02
	6	.00	.00	.00	.00	.01	.02	.04	.04
	7	.00	.00	.00	.01	.02	.04	.08	.11
	8	.00	.00	.01	.01	.02	.04	.11	.36

Table 2.8: Learning outcome with two literals when $s = 4.0$, $P(L_1L_2) = P(\overline{L}_1L_2) = P(L_1\overline{L}_2) = P(\overline{L}_1\overline{L}_2) = 0.25$, $P(Y|L_1L_2) = 0.99$, $P(Y|\overline{L}_1L_2) = 0.9$, $P(Y|L_1\overline{L}_2) = 0.9$, and $P(Y|\overline{L}_1\overline{L}_2) = 0.1$. A double line marks the decision boundary: *Forgotten* vs. *Memorized*.

with precision 0.99, where both *Literal One* and *Literal Two* take part in the rule.

2.3 Rule Interaction

The next and final analysis concerns learning of multiple rules. The **or**-relation between the literals x_1 and x_2 showcases this challenge:

$$x_1 \textbf{ or } x_2. \tag{2.46}$$

You get class Y whenever either x_1, x_2, or both of them are *True*. Otherwise, you get class $\overline{Y}$. Assume that you can only build two rules for class Y. Then only the following pair of rules solve the problem:

R1: if x_1 **then** Y.

R2: if x_2 **then** Y.

When x_1 is *True*, rule **R1** ensures correct output. Rule **R2** takes care of x_2. Note that rule **R3** below also is correct:

			Rule Two			
	$\mathbf{x_2}$		*Forgotten*		*Memorized*	
		$\mathbf{x_1}$	*Forgotten*	*Memorized*	*Forgotten*	*Memorized*
Rule One	*Forg.*	*Forg.*	R1: $\emptyset$ R2: $\emptyset$	R1: $\emptyset$ R2: x_1	R1: $\emptyset$ R2: x_2	R1: $\emptyset$ R2: $x_1 \wedge x_2$
		Mem.	R1: x_1 R2: $\emptyset$	R1: x_1 R2: x_1	**R1: $\mathbf{x_1}$** **R2: $\mathbf{x_2}$**	R1: x_1 R2: $x_1 \wedge x_2$
	Mem.	*Forg.*	R1: x_2 R2: $\emptyset$	**R1: $\mathbf{x_2}$** **R2: $\mathbf{x_1}$**	R1: x_2 R2: x_2	R1: x_2 R2: $x_1 \wedge x_2$
		Mem.	R1: $x_1 \wedge x_2$ R2: $\emptyset$	R1: $x_1 \wedge x_2$ R2: x_1	R1: $x_1 \wedge x_2$ R2: x_2	R1: $x_1 \wedge x_2$ R2: $x_1 \wedge x_2$

Table 2.9: A game between two rules: *Rule One* and *Rule Two*. Each rule has its own memory containing two literals: x_1 and x_2. A literal decides by itself to be *Forgotten* or *Memorized*. The resulting four decisions lead to 16 different rule configurations. The solution configurations for the **or**-relation are marked in bold.

R3: if x_1 and x_2 then Y.

However, you will not be able to capture all the observations that make the **or**-relation *True* if you use rule **R3** in combination with either rule **R1** or rule **R2**.

Two-Rule Game. Again, you get a Literal Game. However, this time the game has four players. Table 2.9 shows the game. Each player decides independently whether to *Forget* or *Memorize* its literal. The two row-players build Rule One (R1) and the two column-players build Rule Two (R2). Column 1/Row 1 covers literal x_2 and Column 2/Row 2 covers literal x_1. With four decision makers, you get 16 different outcomes. Each outcome is a pair of rules. Note that the two outcomes that yield the **or**-relation are in bold.

Tsetlin Machine Memory and Markov Chain. Equip each rule with a four-position memory, shown in Figure 2.17. You build a Markov chain for these two together in the same way that you built a Markov chain for one rule. The new chain gets $(4 \cdot 4)(4 \cdot 4) = 256$ transitions. Building the

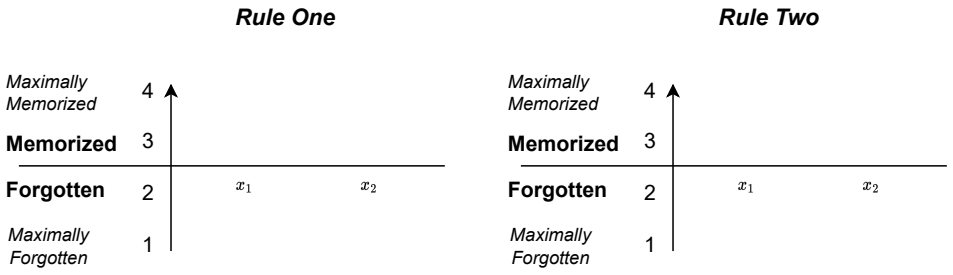

Figure 2.17: Memory of two rules.

chain is left as an exercise. In brief, you go from one rule to two rules in the same way as you go from rolling one to two dice. That is, each die roll is independent so you get the joint probability by multiplication. The probability of rolling two sixes is for instance: $\frac{1}{6} \cdot \frac{1}{6} = \frac{1}{36}$.

No Coordination. The learning outcome when the rules operate without coordination is shown in Table 2.10. The literals are independent and *True* with equal probability: $P(x_1) = P(x_2) = 0.5$. Further, there is 10% output noise: $P(Y|x_1 \vee x_2) = 0.9$ and $P(Y|\neg(x_1 \vee x_2)) = 0.1$. Here, the $\neg$-symbol means logical **not**. Observe how both rule memories end up in memory position 4 for both literals (with the highest probability). That is, you get two instances of rule **R3**. This is a sub-optimal configuration that only captures literal truth values ($x_1 = \mathit{True}, x_2 = \mathit{True}$), but not ($x_1 = \mathit{True}, x_2 = \mathit{False}$) or ($x_1 = \mathit{False}, x_2 = \mathit{True}$).

Coordination With Vote Margin. Finally, add a Voting Margin of value 2. Then the updating of each rule is suppressed based on the number of votes on Y, contrasted against the observed class (Y or $\overline{Y}$). Suppression operates as follows:

- **Zero Votes on Y.** This means that none of the rule conditions are *True*.
  - **Class $\overline{Y}$ observed.** Suppress rule feedback with probability 1.0.
  - **Class Y observed.** Suppress rule feedback with probability 0.0.

			Rule Two															
	$\mathbf{x_2}$		1				2				3				4			
		$\mathbf{x_1}$	1	2	3	4	1	2	3	4	1	2	3	4	4	3	2	1
Rule One	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		4	0	0	0	0	0	0	0	.01	0	0	.01	.01	.01	.01	.01	0
	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		3	0	0	0	0	0	0	0	.01	0	0	.01	.01	.02	.01	.01	0
		4	0	0	0	0	0	0	0	.01	0	0	.01	.02	.05	.02	.01	0
	4	4	0	0	0	0	0	0	0	.01	0	0	.02	.05	.31	.05	.01	0
		3	0	0	0	0	0	0	0	.01	0	0	.01	.02	.05	.02	.01	0
		2	0	0	0	0	0	0	0	.01	0	0	.01	.01	.01	.01	.01	0
		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2.10: Learning outcome with two interacting rules without a Vote Margin. A double line marks the decision boundary: *Forgotten* vs. *Memorized*.

- **One Vote on Y.** This means that a single rule condition is *True*.
  - **Class $\overline{Y}$ observed.** Suppress rule feedback with probability 0.5.
  - **Class Y observed.** Suppress rule feedback with probability 0.5.
- **Two Votes on Y.** This means that both rule conditions are *True*.
  - **Class $\overline{Y}$ observed.** Suppress rule feedback with probability 0.0.
  - **Class Y observed.** Suppress rule feedback with probability 1.0.

This simple modification to the updating of rules completely changes the dynamics of learning. Table 2.11 shows the resulting learning outcome. As you see, now the Tsetlin machine learning produces both:

			Rule Two															
	$\mathbf{x_2}$		1				2				3				4			
		$\mathbf{x_1}$	1	2	3	4	4	3	2	1	4	3	2	1	1	2	3	4
Rule One	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		3	0	0	0	0	0	0	0	0	0	0	0	.01	.04	.01	0	0
		4	0	0	0	0	0	0	0	0	0	0	.01	.04	.16	.04	0	0
	2	4	0	0	0	0	0	0	0	0	0	0	0	.01	.04	.01	0	0
		3	0	0	0	0	0	0	0	0	0	0	0	0	.01	0	0	0
		2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	3	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		2	0	0	0	.01	0	0	0	0	0	0	0	0	0	0	0	0
		1	0	0	.01	.04	.01	0	0	0	0	0	0	0	0	0	0	0
	4	1	0	0	.04	.16	.04	.01	0	0	0	0	0	0	0	0	0	0
		2	0	0	.01	.04	.01	0	0	0	0	0	0	0	0	0	0	0
		3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 2.11: Learning outcome with two interacting rules with Vote Margin. A double line marks the decision boundary: *Forgotten* vs. *Memorized*.

R1: if x_1 **then** Y.

R2: if x_2 **then** Y.

and the alternative solution:

R1: if x_2 **then** Y.

R2: if x_1 **then** Y.

Any sub-optimal solution involving rule **R3** occurs with zero probability. Hence, the power of the scheme!

2.4 Summary

Here are the main points from this chapter:

- Single-literal learning in Tsetlin machines is a stochastic process that takes the form of a *Markov chain*. The Markov chain mixes Recognize, Erase, and Reject feedback based the classification problem at hand. Specifying the Markov chain, you obtain formulas that describe the learning outcome exactly. You can also approximate the learning outcome by means of the transition matrix of the chain.
- The outcome of single-literal learning reveals how the Tsetlin machine picks up frequent literals, emphasizing literals that distinguish between objects of different classes. The outcome also shows that learning gets more accurate with increasing memory depth.
- When you introduce several literals, you get a game between literals. That is, each literal is a decision-maker that decides whether to take part in the rule at hand. A game matrix specifies the outcome of this game, yielding a more complex Markov chain that captures literal interaction. The multi-literal learning outcome shows how the Tsetlin machine selects the best literals out of many and combines multiple for increased precision.
- With multiple rules, each rule gets its own team of decision-makers. These teams jointly construct all of the rules in every round of the game. The resulting stochastic process covers how rule interaction emerges from literal interaction. Furthermore, the process shows how the Voting Margin helps the Tsetlin machine avoid sub-optimal configurations, producing optimal ones instead.
- Overall, Tsetlin machine learning moves towards a stationary probability distribution over rule configurations. The stationary distribution is decided by the classification problem itself, the memory depth, the *Forget Value*, and the *Memorize Value*, but not the initialization happening at start of learning. This means that Tsetlin machines learn continuously. The end-point of learning is simultaneously the starting point for further learning.