Markov Chains

Lucas Ondel

June 30, 2022

Contents

1 Semiring

A semiring $K = (\mathbb{K}, +_K, \times_K, 0_K, 1_K)$ is a set $\mathbb{K}$ equipped with a commutative binary operation $+_K$ (called "addition") with identity element 0_K and an associative binary operation $\times_K$ (called "multiplication") with identity element 1_K that distributes over $+_K$. Note that we drop the K subscript when there is no ambiguity.

Here are commonly used semirings:

Semiring name	$\mathbb{K}$	$x +_K y$	$x \times_K y$	$x /_K y$	0_K	1_K
Boolean	$\{\text{true}, \text{false}\}$	$x \vee y$	$x \wedge y$		false	true
Log	$\mathbb{R}$	$\ln(e^x + e^y)$	$x + y$	$x - y$	$-\infty$	0
Probability	$\mathbb{R}^+$	$x + y$	$x \cdot y$	x/y	0	1
Tropical	$\mathbb{R}$	max(x, y)	$x + y$	$x - y$	$-\infty$	0
Union-Concatenation	$\{S : S \subseteq \Sigma^*\}$	$x \cup y$	$\{ab : a \in x, b \in y\}$		$\{\}$	$\{\epsilon\}$

Here are the main properties of the semirings:

Semiring name	idempotent	zero-sum free	divisible	ordered
Boolean	✓	✓		
Log		✓	✓	✓
Probability		✓	✓	✓
Tropical	✓	✓	✓	✓
Union-Concatenation	✓	✓		

2 Finite State Machines

We provide here a formal definition of (weighted) Finite State Machines (FSMs). This definition slightly differs from the traditional formalism of finite automata. These changes are introduced in order to present simple and yet powerful framework for structured inference.

2.1 Definitions

We define a FSM $\mathcal{M}$ by the tuple $\mathcal{M} = (Q, \Sigma, K, L, \boldsymbol{\alpha},\ \mathbf{T},\ \boldsymbol{\omega}, \boldsymbol{\lambda})$ where:

- $Q = \{1, \dots, d\}$ is the set of states (with cardinality d) identified as integers
- Σ is a set of symbols
- K is an zero-sum free and ordered semiring for the FSM's weights
- $L = (\Sigma \cup \{\epsilon\}, \times_L)$ is a free monoid
- $\boldsymbol{\alpha} \in K^d$ is a vector such that α_i is the initial weight of the state $i \in Q$
- $\mathbf{T} \in K^{d \times d}$ is a matrix such that T_{ij} is the transition weight from the state $i \in Q$ to the state $j \in Q$
- $\boldsymbol{\omega} \in K^d$ is a vector such that ω_i is the final weight of the state $i \in Q$
- $\boldsymbol{\lambda} \in L^d$ is a vector of symbol such that λ_i is the symbol of the state $i \in Q$.

An example of FSM with its graphical and matrix representation is shown in Figure 1.

initial and final states: We say that a state i is an *initial state* if its initial weight is greater than 0, i.e $\alpha_i > 0$. Similarly, we say that a state i is a *final state* if its final weight is greater than 0, i.e. $\omega_i > 0$.

path: A path

$$\boldsymbol{\pi} = (\pi_1, \pi_2, \dots, \pi_n), \quad \pi_i \in Q, \quad \forall i \in \{1, \dots, n\} \tag{1}$$

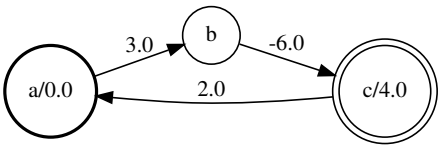

Figure 1: example of FSM in the log-semiring where:

$$Q = \{1, 2, 3\} \qquad \Sigma = \{a, \ldots, z\} \tag{4}$$

$$\boldsymbol{\alpha} = \begin{bmatrix} 0 \\ -\infty \\ -\infty \end{bmatrix} \qquad \boldsymbol{\omega} = \begin{bmatrix} -\infty \\ -\infty \\ 4 \end{bmatrix} \tag{5}$$

$$\mathbf{T} = \begin{bmatrix} -\infty & 3 & -\infty \\ -\infty & -\infty & -6 \\ 2 & -\infty & -\infty \end{bmatrix} \qquad \boldsymbol{\lambda} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \tag{6}$$

is a sequence of states. The weight of a path $\boldsymbol{\pi}$ is given by the function $\mu : Q^n \to K$:

$$\mu(\boldsymbol{\pi}) = \alpha_{\pi_1} \Big[\prod_{i=2}^{n-1} T_{\pi_{i-1}, \pi_i} \Big] \omega_{\pi_N}. \tag{2}$$

Similarly, the label sequence of the path $\boldsymbol{\pi}$ is given by the function $\sigma : Q^n \to L$:

$$\sigma(\boldsymbol{\pi}) = \prod_{i=1}^{n} \lambda_{\pi_i}. \tag{3}$$

input sequence: We define an input sequence $\mathbf{s}$ to a FSM as a sequence of labels:

$$\mathbf{s} = \prod_{i=1}^{n} s_i, \quad s_i \in L. \tag{7}$$

The weight of an input sequence is given by the function $\nu : L \to K$:

$$\nu(\mathbf{s}) = \sum_{\mathbf{x} \in P} \mu(\mathbf{x}), \tag{8}$$

where $P = \{\boldsymbol{\pi} : \ \sigma(\boldsymbol{\pi}) = \mathbf{s}, \ \boldsymbol{\pi} \in \mathcal{M}\}$ is the set of paths from $\mathcal{M}$ with label sequence $\mathbf{s}$. We say that an input sequence is *accepted by* $\mathcal{M}$ if $\nu(\mathbf{s}) > 0$. We denote S the set of accepted input sequence of a FSM, i.e. $S = \{\mathbf{s} : \nu(\mathbf{s}) > 0\}$.

2.2 Total sum

We define the total weight sum of a FSM ζ as the sum of all the weights of its accepted input sequences:

$$\zeta = \sum_{\mathbf{s} \in S} \nu(\mathbf{s}). \tag{9}$$

The total weight sum can be estimated efficiently via dynamic programming. Let be ζ_n the *partial total sum* of a FSM $\mathcal{M}$ defined as the sum of all the weights of accepted input sequence of size smaller of equal to n:

$$\zeta_n = \sum_{\mathbf{s} \in \{\mathbf{s}:\ |\mathbf{s}| \leq n,\ \mathbf{s} \in S\}} \nu(\mathbf{s}). \tag{10}$$

ζ_n can be calculated through the following recursion:

$$\mathbf{v}_n = \mathbf{T}^\top \mathbf{v}_{n-1} \tag{11}$$

$$\zeta_n = \boldsymbol{\omega}^\top \mathbf{v}_n, \tag{12}$$

where the recursion is initialized with $\mathbf{v}_1 = \boldsymbol{\alpha}$

3 FSM operations

We describe here some operations over FSMs. A FSM $\mathcal{M}_i$ is defined as:

$$\mathcal{M}_i = (Q_i, \Sigma_i, K_i, L_i, \boldsymbol{\alpha}_i, \mathbf{T}_i, \boldsymbol{\omega}_i, \boldsymbol{\lambda}_i). \tag{13}$$

The path weight, path label and input sequencs weight function associated to this FSM are denoted $\mu_i(\cdot)$, $\sigma_i(\cdot)$ and $\nu_i(\cdot)$ respectively. The set of input sequences accepted by $\mathcal{M}_i$ is $S_i = \{\mathbf{s} : \nu_i(\mathbf{s}) > 0\}$. In the following, for binary operator over 2 FSMs, we assume tacitly that both FSMs have the same set of symbols, i.e. $\Sigma = \Sigma_1 = \Sigma_2$ and the same weight semiring, i.e. $K = K_1 = K_2$, and the same label free monoid, i.e. $L = L_1 = L_2$.

3.1 Renormalization

We say that a FSM $\mathcal{M}$ is *normalized* if (i) the sum of its intial weights sum up to 1, i.e. $\sum_{i \in Q} \alpha_i = 1$, and (ii) the sum of the transition weights from a state i to all other states and the final weight of the state i sum up to 1, i.e. $\omega_i + \sum_{j \in Q} T_{ij} = 1$. A FSM $\mathcal{M}_1$ with a zero-sum free and divisble weight semiring can be transformed into normalized FSM $\mathcal{M}_2$ via the renormalization operation: $\mathcal{M}_2 = \text{renorm}(\mathcal{M}_1)$. The resulting FSM is obtained by the following

construction:

$$Q_2 = Q_1 \qquad \boldsymbol{\alpha}_2 = \Big(\sum_i \alpha_{1i}\Big)^{-1} \boldsymbol{\alpha}_1 \tag{14}$$

$$\mathbf{T}_2 = \begin{bmatrix} (\omega_1 + \sum_i T_{1,1i})^{-1}\mathbf{T}_{1,1} \\ \vdots \\ (\omega_d + \sum_i T_{1,di})^{-1}\mathbf{T}_{1,d} \end{bmatrix} \qquad \boldsymbol{\omega}_2 = \begin{bmatrix} (\omega_1 + \sum_i T_{1,1i})^{-1} \\ \vdots \\ (\omega_d + \sum_i T_{1,di})^{-1} \end{bmatrix} \odot \boldsymbol{\omega}_1 \tag{15}$$

$$\boldsymbol{\lambda}_2 = \boldsymbol{\lambda}_1 \tag{16}$$

3.2 Union

The union of two FSMs $\mathcal{M}_1 \cup \mathcal{M}_2$ gives a FSM $\mathcal{M}_3$ such that $S_3 = S_1 \cup S_2$ and $\forall \mathbf{s} \in S_3$, $\nu_3(\mathbf{s}) = \nu_1(\mathbf{s}) + \nu_2(\mathbf{s})$. $\mathcal{M}_3$ can be obtained with the following construction:

$$Q_3 = \{1, \ldots, |Q_1| + |Q_2|\} \qquad \boldsymbol{\alpha}_3 = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \boldsymbol{\alpha}_2 \end{bmatrix} \tag{17}$$

$$\mathbf{T}_3 = \begin{bmatrix} \mathbf{T}_1 & \\ & \mathbf{T}_2 \end{bmatrix} \qquad \boldsymbol{\omega}_3 = \begin{bmatrix} \boldsymbol{\omega}_1 \\ \boldsymbol{\omega}_2 \end{bmatrix} \tag{18}$$

$$\boldsymbol{\lambda}_3 = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \end{bmatrix} \tag{19}$$

3.3 Concatenation

The concatenation of two FSMs $\text{concat}(M_1, \mathcal{M}_2)$ gives a FSM $\mathcal{M}_3$ such that $S_3 = \{\mathbf{s}_1\mathbf{s}_2 : \mathbf{s}_1 \in S_1,\ \mathbf{s}_2 \in S_2\}$. $\mathcal{M}_3$ can be obtained with the following construction:

$$Q_3 = \{1, \ldots, |Q_1| + |Q_2|\} \qquad \boldsymbol{\alpha}_3 = \begin{bmatrix} \boldsymbol{\alpha}_1 \\ 0\boldsymbol{\alpha}_2 \end{bmatrix} \tag{20}$$

$$\mathbf{T}_3 = \begin{bmatrix} \mathbf{T}_1 & \boldsymbol{\omega}_1\boldsymbol{\alpha}_2^\top \\ & \mathbf{T}_2 \end{bmatrix} \qquad \boldsymbol{\omega}_3 = \begin{bmatrix} 0\boldsymbol{\omega}_1 \\ \boldsymbol{\omega}_2 \end{bmatrix} \tag{21}$$

$$\boldsymbol{\lambda}_3 = \begin{bmatrix} \boldsymbol{\lambda}_1 \\ \boldsymbol{\lambda}_2 \end{bmatrix} \tag{22}$$

3.4 Reversal

The reversal (denoted $^\top$) of a FSM $\mathcal{M}_1$ yields a FSM $\mathcal{M}_2 = \mathcal{M}_1^\top$ such that $S_2 = \{\overleftarrow{\mathbf{s}} : \mathbf{s} \in S_1\}$ where $\overleftarrow{\mathbf{s}}$ is the sequence $\mathbf{s}$ in reversed order. $\mathcal{M}_2$ is obtained by the following construction:

$$Q_2 = Q_1 \qquad \boldsymbol{\alpha}_2 = \boldsymbol{\omega}_1 \tag{23}$$

$$\mathbf{T}_2 = \mathbf{T}_1^\top \qquad \boldsymbol{\omega}_2 = \boldsymbol{\alpha}_1 \tag{24}$$

$$\boldsymbol{\lambda}_2 = \boldsymbol{\lambda}_1 \tag{25}$$

3.5 Composition

Let's consider a FSM $\mathcal{M}_1$ with d states. We would like to compose $\mathcal{M}_1$ with a sequence of d FSMs $\mathcal{M}^{1:d} = (\mathcal{M}^1, \ldots, \mathcal{M}^d)$ That is to say that the ith state of $\mathcal{M}_1$ is replaced with the FSM $\mathcal{M}^i$. The composition of $\mathcal{M}_1$ and $\mathcal{M}^{1:d}$, denoted $\mathcal{M}_1 \circ \mathcal{M}^{1:d}$, yields a FSM $\mathcal{M}_2$ which can be obtained with the following construction:

$$Q_2 = \{1, \ldots, \sum_{i=1}^{d} |Q^i|\} \tag{26}$$

$$\boldsymbol{\alpha}_2 = \begin{bmatrix} \alpha_{1,1}\boldsymbol{\alpha}^1 \\ \vdots \\ \alpha_{1,d}\boldsymbol{\alpha}^d \end{bmatrix} \tag{27}$$

$$\mathbf{T}_2 = \begin{bmatrix} \mathbf{T}^1 & & \\ & \ddots & \\ & & \mathbf{T}^d \end{bmatrix} + (\mathbf{M}_K \mathbf{T}_1 \mathbf{M}_K^\top) \odot \left(\begin{bmatrix} \boldsymbol{\omega}_1 \\ \vdots \\ \boldsymbol{\omega}_d \end{bmatrix} \begin{bmatrix} \boldsymbol{\alpha}_1 \\ \vdots \\ \boldsymbol{\alpha}_1 \end{bmatrix}^\top \right) \tag{28}$$

$$\boldsymbol{\omega}_2 = \begin{bmatrix} \omega_{1,1}\boldsymbol{\omega}^1 \\ \vdots \\ \omega_{1,d}\boldsymbol{\omega}^d \end{bmatrix} \tag{29}$$

$$\boldsymbol{\lambda}_2 = \mathbf{M}_L^\top \begin{bmatrix} \lambda_{1,1}\boldsymbol{\lambda}^1 \\ \vdots \\ \lambda_{1,d}\boldsymbol{\lambda}^d \end{bmatrix}, \tag{30}$$

$\mathbf{M}_K$ is a matrix whose elements belong to the semiring K and it is defined as:

$$\mathbf{M}_K = \begin{bmatrix} \mathbf{1}_{|Q_1|} & & & \\ & \mathbf{1}_{|Q_2|} & & \\ & & \ddots & \\ & & & \mathbf{1}_{|Q_d|} \end{bmatrix}. \tag{31}$$

where $\mathbf{1}_{|Q_i|}$ is a vector of 1 of size $|Q_i|$. $\mathbf{M}_L$ is defined identically but has elements in the semiring L.

3.6 Weight propagation

Propagate the weights through the FSM. This operation is needed before determinization to guarantee that the output FSM will be equivalent to the input one.

3.7 Determinization

Standard powerset construction algorithm.

4 Markov chains

We introduce here the concept of Markov chains and show it relations with FSMs.

4.1 Definition

A (discrete-time) Markov chain is a stochastic process for which the joint distribution of N consecutive samples $\mathbf{z} = (z_1, \ldots, z_N)^\top$ factorizes as:

$$p(\mathbf{z}) = p(z_1) \prod_{i=2}^{n} p(z_i|z_{i-1}), \tag{32}$$

where $p(z_1)$ is the *initial probability* and $p(z_i|z_{i-1})$ is the *transition probability*. Let's consider a slightly revise definition of this stochastic process: let suppose that, in order to observe a sequence $\mathbf{z}$ the stochastic process has to halt for an instant. Let's assume further that the probability of halting depends on the state. The process can halt when certain states are drawn but not for some other states. Noting $p(\emptyset|z_n)$ the probability of halting when in state z_n, the probability of observing a sequence becomes:

$$p(\mathbf{z}) = p(z_1)p(\emptyset|z_n) \prod_{i=2}^{n} p(z_i|z_i - 1) \tag{33}$$

4.2 FSM representation

Equation 33 can be very naturally mapped to a d-state FSM using the probability semiring with:

$$\boldsymbol{\alpha} = \begin{bmatrix} p(z_1 = 1) \\ \vdots \\ p(z_1 = d) \end{bmatrix} \tag{34}$$

$$\mathbf{T} = \begin{bmatrix} p(z_i = 1|z_{i-1} = 1) & \ldots & p(z_i = d|z_{i-1} = 1) \\ \vdots & \ddots & \vdots \\ p(z_i = 1|z_{i-1} = d) & \ldots & p(z_i = d|z_{i-1} = d) \end{bmatrix} \tag{35}$$

$$\boldsymbol{\omega} = \begin{bmatrix} p(\emptyset|z_1 = 1) \\ \vdots \\ p(\emptyset|z_1 = d) \end{bmatrix}. \tag{36}$$

The states'label is application dependent and can be defined in many ways.

The relation between the Markov chains and FSM brings many advantages, notably:

- the constructive operations (union, concatenation, composition, ...) allow to create easily complex distribution over sequences
- several important inference algorithms can be seen as special instance of the total sum algorithm, there are therefore easily implemented and optimized.

4.3 Inference algorithms

The use Markov chains in machine learning problem usually necessitate to solve two inference problem: (i) state-marginalization and (ii) calculate the probability of the most likely sequence. We show how this two algorithms can be calculated easily using the FSM formalism.

4.3.1 State marginalization

We aim to calculate the following quantity:

$$p(z_i) = \sum_{z_1,\dots,z_n} p(z_1, \dots, z_{i-1}, z_i, \dots, z_n) \tag{37}$$

$$= \sum_{z_{i-1}} p(z_i|z_{i-1})p(z_{i-1}). \tag{38}$$

Equation 38 leads to a recursive function which is trivially evaluated with the total sum algorithm:

$$\mathbf{v}_i = \begin{bmatrix} p(z_i = 1) \\ \vdots \\ p(z_i = d) \end{bmatrix} = \mathbf{T}^\top \mathbf{v}_{i-1} \tag{39}$$

with initial condition $\mathbf{v}_1 = \boldsymbol{\alpha}$.

In practice, (39) is not numerically stable as it multiplies many numbers smaller than ones leading to numerical underflow of floating point arithmetic. However, this issue is easily solved by taking the logarithm of all values of $\mathbf{T}$ and $\mathbf{v}_i$ and using the log-semiring.

This computation is at the heart of the forward-backward algorithm used to train Hidden Markov Models and to compute the gradient of several sequence-discriminative objective function.

4.3.2 Best path probability

We aim to find:

$$\max_{z_1,\dots,z_n} p(\mathbf{z}) = \max_{z_{i+1},\dots,z_n} p(z_{i+1}, \dots, z_n) \tag{40}$$

Whereas the graph conveniently represents the possible trajectories in the state-space, the transition matrix $\mathbf{T}$ allows to express state marginalization as a matrix-vector multiplication. For instance:

$$p(z_n) = \sum_{i \in \{a,b,c\}} p(z_{n-1} = i, z_n) \tag{41}$$

$$= \sum_{i \in \{a,b,c\}} p(z_n | z_{n-1} = i) p(z_{n-1} = i) \tag{42}$$

$$\mathbf{v}_n = \mathbf{T}\mathbf{v}_{n-1} \tag{43}$$

, where:

$$\mathbf{v}_n = \begin{bmatrix} p(z_n = a) \\ p(z_n = b) \\ p(z_n = c) \end{bmatrix}. \tag{44}$$

Equation (43) is the "core" operation of many Markov chains related algorithm such as forward-backward (for training models) and viterbi (for decoding speech). For Markov chains that have a large number of states, this operation is problematic as its complexity is quadratic in the number of states: $\mathcal{O}(2D^2)$ where D is the number of states. The rest of the document describes how to exploit the structure of the Markov chains to decrease the complexity of this operation.

4.4 Compact graphical form

In many application, the transition probabilities have some structure allowing to represent the Markov chain in a more compact manner. For instance, let's consider the following transition probabilities:

$$p(z_n = j | z_{n-1} = i) = \begin{cases} \gamma + \nu_i \delta_j & \text{if } i = a \text{ and } j = b \\ \nu_i \delta_j & \text{otherwise.} \end{cases} \tag{45}$$

Defined in this way, this Markov chain has $2 \cdot 3 + 1 = 7$ parameters instead of $3 \cdot 3 = 9$ in the general case. The graphical representation of this constrained Markov chain is shown in Fig. 2.

4.5 Efficient marginalization

In many applications, we would like to use the structure of the Markov chain to efficiently marginalize over a state. The formula in (43) can be prohibitive to evaluate if the state-space is large. The idea is to use the constraints of the Markov chain to efficiently calculate the matrix-vector product.

In our particular example, observe that the transition matrix can be written as:

$$\mathbf{T} = \mathbf{S} + \boldsymbol{\nu}\boldsymbol{\delta}^\top, \tag{46}$$

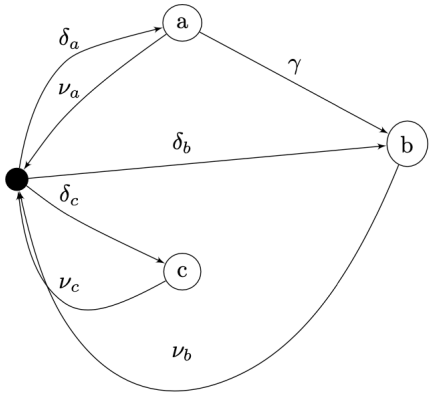

Figure 2: Graphical representation of a constrained Markov chain. The filled node is a "phony" state equivalent of the *epsilon-arc* in the WFST framework.

where:

$$\mathbf{S} = \begin{bmatrix} 0 & \gamma & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \boldsymbol{\nu} = \begin{bmatrix} \nu_a \\ \nu_b \\ \nu_c \end{bmatrix} \quad \boldsymbol{\delta} = \begin{bmatrix} \delta_a \\ \delta_b \\ \delta_c \end{bmatrix}. \tag{47}$$

Consequently, we have:

$$\mathbf{T}\mathbf{v}_{n-1} = (\mathbf{S} + \boldsymbol{\nu}\boldsymbol{\delta}^\top)\mathbf{v}_{n-1}, \tag{48}$$

and using the associativity and the distributive properties of the addition and multiplication we re-write it as:

$$\mathbf{T}\mathbf{v}_{n-1} = \mathbf{S}\mathbf{v}_{n-1} + \boldsymbol{\nu}(\boldsymbol{\delta}^\top \mathbf{v}_{n-1}). \tag{49}$$

Calculating the matrix-vector product following the operation order of (49), the complexity reduces to: $\mathcal{O}(2Q+2D)$ where Q is the number of non-zero elements in $\mathbf{S}$.

Remark: in the general case, it is easy to show that:

$$\mathbf{T} = \mathbf{S} + \sum_{k}^{K} \boldsymbol{\nu}_k \boldsymbol{\delta}_k^\top, \tag{50}$$

where K is the number of "phony" states in the graphical representation of the Markov chain[1][2].

[1]Here, I assume that there is no looping path starting from a "phony" state that does not contain a "real" state. This constraint is necessarily met in practice.

[2]The factorization in (50) is not unique.