

Composable Markov Building Blocks

Sander Evers, Maarten M. Fokkinga, and Peter M.G. Apers

University of Twente

Abstract. In situations where disjunct parts of the same process are described by their own first-order Markov models, these models can be joined together under the constraint that there can only be one activity at a time, i.e. the activities of one model coincide with non-activity in the other models. Under certain conditions, nearly all the information to do this is already present in the component models, and the transition probabilities for the joint model can be derived in a purely analytic fashion. This provides a theoretical basis for building scalable and flexible models for sensor data.

1 Introduction

In order to deal with time series of sensor data, it is useful to have a statistical model of the observed process. This helps to smooth noisy readings, or to detect faulty observations, by keeping track of in what states the process is most likely to be. For example, in object localization, if we have a sequence of position observations 3–2–4–18–5–4 we know we can disregard the 18 reading because the model assigns a very low likelihood to the object moving back and forth so fast. The parameters of such a statistical model can be obtained from domain expert knowledge or by (supervised or unsupervised) *learning* from data.

When the state space of a statistical model is large and heterogeneous, these parameters become hard to obtain. Therefore, like in all large and complex problems, it is fruitful to look for *composability* of statistical models; composability is often the key to flexibility and scalability. In this article, we consider a specific opportunity for composability, where several *disjunct parts* of the state space can be described by their own first-order Markov models (we have investigated first-order Markov models because these are the most simple). We present a mathematical result about the conditions under which these models are composable, and the method to perform this composition.

In order to illustrate this result, we use a running example about activity recognition, where the heterogeneity of the state space stems from the fact that different types of sensors are used for several subclasses of activities. This particular example actually has a very small state space, and we stress the fact that it is used only for illustration of the mathematical procedure; it is not meant as a realistic application.

Fig. 1a shows our example, which consists of three component models *body*, *object* and *computer*, which are associated with three different types of sensors:

- Motion sensors on the body are used to classify the activities walking and climbing stairs.
- Sensors on coffee cup and a book register interaction with these objects, and are used to classify coffee drinking and book reading.
- From desktop computer interactions, the activities logging in, reading mail, reading an article and writing a document are classified.

Each model is first-order Markovian and also contains an additional state of *non-activity*, in which the monitored person is considered to be doing ‘something else’, which cannot be observed more precisely in that model. Our goal is to compose these models together under the constraint that *there can only be one activity at a time*, i.e. the activities of one model coincide with non-activity in the other models.

We have investigated the conditions under which this model composition can happen in a purely analytic fashion, without assuming or adding any other information apart from the *structure* of the transition graph between the component models (Fig. 1b). The main result is that when this structure is ‘sparse enough’, all inter-component transition probabilities can be deduced from the component models. The technique that we present checks this condition and deduces the probabilities, and is novel to the best of our knowledge.

The remainder of this article is structured as follows. In Sect. 2, we formalize the problem; Sect. 3 turns it into a set of linear equations; in Sect. 4 we present a specialized method to solve this system; in Sect. 5 we apply this method to our problem; Sect. 6 concludes.

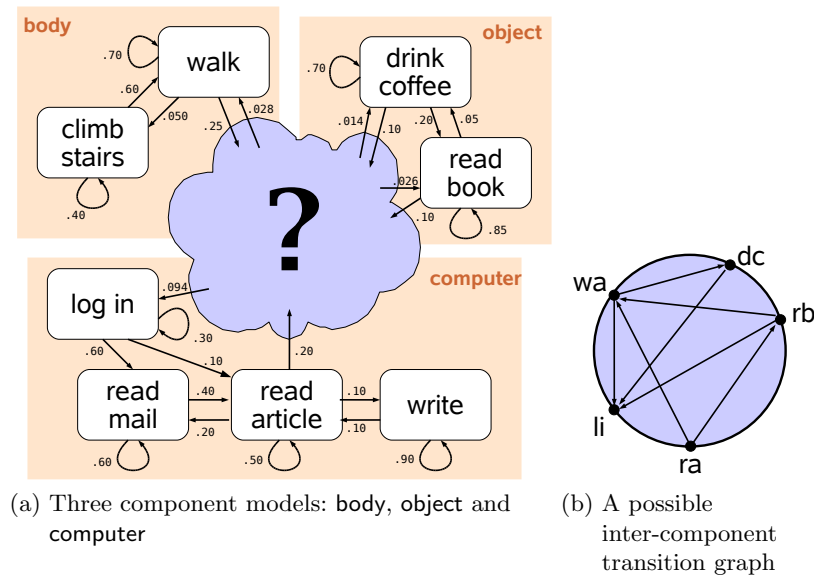


Fig. 1: Running example: activity recognition

2 Markov Models and Pseudo-Aggregated Components

In this section, we give a quick introduction to Markov models, and we formalize the relation between a global Markov model G and its components C_1, \dots, C_k .

Consider a process that can be in one of the states of a finite set S_G at each subsequent point in (discrete) time. One of the simplest and most used models that probabilistically relates the states in such a series to each other is the *homogeneous first-order Markov model*. Informally, *first-order Markov* means that the state at time t (denoted X_t) is only dependent on the state at time $t-1$ (X_{t-1}), i.e. for guessing the next state, knowledge of the current state obsoletes all knowledge of previous states.

The parameters of a first-order Markov model consist of the conditional probabilities $P(X_t = j | X_{t-1} = i)$ with $i, j \in S_G$. We consider *homogeneous* models, which means that these conditional probabilities are the same for all values of t . This allows one to specify all these parameters using a transition function G , in which $G(i, j) = P(X_t = j | X_{t-1} = i)$. Often, this function is represented by a matrix, for which we will also use the symbol G ; furthermore, we will also use G to refer to the corresponding Markov model itself. In the remainder of this article, when we mention a Markov model, a first-order heterogeneous Markov model is implied.

The probabilities in a Markov model can be interpreted as observation frequencies. An observation of a model consists of a consecutive sequence of states; when these observations become longer, the number of transitions from i to j (i.e. $[i, j]$ subsequences) divided by the number of transitions from i (i.e. $[i, _]$ subsequences, or almost equivalently, the number of i occurrences) would converge to $G(i, j)$. Learning a model from the data works the other way around: the $G(i, j)$ parameters are estimated by using the observed frequencies.

Now, consider the situation in which we assume a Markov model G with states $\{1, \dots, n\}$, but cannot distinguish between the states m through n in our observations (with $m < n$). When we use the observed transitions to estimate a Markov model, we arrive at a different Markov model C with $|S_C| = n$ states. This model is called the *pseudo-aggregation* of G with respect to the partition $S_C = \{\{1\}, \{2\}, \dots, \{m-1\}, \{m, \dots, n\}\}$. (In the formal definition of pseudo-aggregation[1], the parameters of the pseudo-aggregated model C are directly defined in terms of the G parameters, i.e. without referring to observations.)

The goal of this article is to construct an unknown global Markov model G (in our running example, $S_G = \{\text{cs}, \text{wa}, \text{dc}, \text{rb}, \text{li}, \text{rm}, \text{ra}, \text{wr}\}$; each activity is abbreviated to two letters) from several known pseudo-aggregations C_1, \dots, C_k which have a special form:

- Each partition S_{C_i} consists of one or more singleton states and exactly one non-singleton state. In our running example:
 - $S_{\text{body}} = \{\{\text{cs}\}, \{\text{wa}\}, \{\text{dc}, \text{rb}, \text{li}, \text{rm}, \text{ra}, \text{wr}\}\}$,
 - $S_{\text{object}} = \{\{\text{dc}\}, \{\text{rb}\}, \{\text{cs}, \text{wa}, \text{li}, \text{rm}, \text{ra}, \text{wr}\}\}$, and
 - $S_{\text{computer}} = \{\{\text{li}\}, \{\text{rm}\}, \{\text{ra}\}, \{\text{wr}\}, \{\text{cs}, \text{wa}, \text{dc}, \text{rb}\}\}$.

- Each state s from S_G corresponds to a singleton state $\{s\}$ in exactly one of the S_{C_i} partitions. We say that the state *belongs to* a specific model C_i . The model to which state s belongs is written $\llbracket s \rrbracket$. So, $\llbracket cs \rrbracket = \text{body}$.

For clarity of exposition we will hereafter slightly transcend this formalization and identify the singleton states with their only member, so cs can actually mean $\{cs\}$.

If the states $i, j \in S_G$ belong to the same component model ($\llbracket i \rrbracket = \llbracket j \rrbracket$), we will call the transition from i to j an *intra-component* transition. Otherwise ($\llbracket i \rrbracket \neq \llbracket j \rrbracket$), it is called an *inter-component* transition. A state that is involved in at least one inter-component transition is called a *border state*. The directed graph consisting of all border states as vertices and all *possible* inter-component transitions (i, j) as edges (i.e. $G(i, j) > 0$) is called the *inter-component transition graph*.

In the remainder of the article, we will show that it is possible to completely reconstruct G under the conditions that:

- we know the inter-component transition graph, i.e. we know which inter-component transitions are possible (have $G(i, j) > 0$).
- the inter-component transition graph is fully connected and does not contain direction-alternating cycles (a concept that we will explain in section 4).

In our example, the border states are $\{\text{wa,dc,rb,li,ra}\}$. An example inter-component transition graph is shown in Fig. 1b. Note that it is not derived from the component models; it is extra information that we add.

3 Transforming to the Domain of Long-Run Frequencies

In the global model G , the transition probabilities $G(i, j)$ for intra-component transitions can be taken directly from the C model to which the states belong: $G(i, j) = \llbracket i \rrbracket(i, j)$. The problem lies with the inter-component transitions: we know for which transitions $G(i, j) > 0$, but we don't know the exact values. However, under certain conditions the C models already contain enough information to deduce them in a completely analytic fashion. To do this, we transform the problem from the domain of conditional probabilities $G(i, j)$ into (unconditional) long-run frequencies $F(i, j)$. In Sect. 5, we will transform the solution back.

The unconditional long-run frequency $F(i, j)$ is the frequency with which a transition from i to j would occur compared to the *total* number of transitions (instead of to the transitions from i). To transform conditional frequencies into unconditional frequencies, we need to know the proportion of the total time spent in each state. These proportions are known[2] to be equal to the stationary distribution π_G , which is the normalized left eigenvector of the transition matrix G with eigenvalue 1, i.e. the solution to

$$\begin{aligned} \pi_G \cdot G &= \pi_G \\ \sum_i \pi_G(i) &= 1 \end{aligned}$$

whose existence and uniqueness is guaranteed when the chain is irreducible and ergodic. (We will come back to these notions in Sect. 5.) It is well known how to calculate a stationary distribution, but we cannot calculate π_G directly because we do not know the complete G matrix. Instead, we use the stationary distributions of the C_i matrices; it is known ([1], Lemma 4.1) that these are equivalent to π_G , up to aggregation. For each state i , we use the model $\llbracket i \rrbracket$ to which it belongs:

$$\pi_G(i) = \pi_{\llbracket i \rrbracket}(i)$$

Using this, we can simply calculate $F(i, j)$ for all *intra*-component transitions from $\llbracket i \rrbracket(i, j)$ by multiplying it with proportion of time spent in i :

$$F(i, j) = \pi_G(i) \cdot \llbracket i \rrbracket(i, j)$$

We need to solve $F(i, j)$ for *inter*-component transitions. From the inter-component transition graph, we know which of these frequencies are 0. We solve the rest of them by equating, for each border state i , the summed frequencies of incoming transitions (including self-transitions) to the proportion of time spent in i (because every time unit spent in i is preceded by a transition to i), and doing the same for the outgoing transitions (because every time unit spent in i is followed by a transition from i):

$$\begin{aligned} \sum_h F(h, i) &= \pi_G(i) \\ \sum_j F(i, j) &= \pi_G(i) \end{aligned}$$

We can move all the known quantities to the right-hand side:

$$\begin{aligned} \sum_{h|\llbracket h \rrbracket \neq \llbracket i \rrbracket} F(h, i) &= \pi_G(i) - \sum_{h|\llbracket h \rrbracket = \llbracket i \rrbracket} F(h, i) \\ \sum_{j|\llbracket i \rrbracket \neq \llbracket j \rrbracket} F(i, j) &= \pi_G(i) - \sum_{j|\llbracket i \rrbracket = \llbracket j \rrbracket} F(i, j) \end{aligned}$$

We are then left with a system of linear equations, with twice as much equations as there are border states, and as much unknowns as there are inter-component transitions. In principle, we could solve these equations using a standard method such as Gauss-Jordan elimination, but in the next section we present a technique that is tailored to the special structure of this system. It has the benefit that it directly relates the conditions under which the system has one unique solution to the inter-component transition graph, and that it checks these conditions (and solves the equations) in a time proportional to the number of inter-component transitions.

4 Distributing Vertex Sum Requirements

In this section, we abstract from the Markov model problem, and present a method for solving a set of linear equations associated with a directed graph: each edge corresponds to an unknown, and each vertex corresponds to two equations (regarding incoming and outgoing edges).

Given a directed graph $G = (V, E)$, with vertex set V and edge set $E \subseteq V \times V$, and two vertex sum requirements $f^+, f^- : V \rightarrow \mathbb{R}$, which specify for each vertex the sum of the weights on its outgoing and incoming edges, respectively (loops count for both), the goal is to find a weight distribution $f : E \rightarrow \mathbb{R}$ matching these requirements, i.e.

$$f^+(v) = \sum_{w|(v,w) \in E} f(v,w)$$

$$f^-(v) = \sum_{u|(u,v) \in E} f(u,v)$$

for all $v \in V$. In this section, we present a necessary and sufficient condition on the structure of G for the uniqueness of such a distribution, and an algorithm to determine it (if it exists at all). For the proof and algorithm, we use an undirected representation of G , which we call its uncoiled graph.

Definition 1. *Given directed graph $G = (V, E)$ with n vertices $\{v_1, v_2, \dots, v_n\}$, we define its uncoiled graph $U = (S + T, E')$. U is an undirected bipartite graph with partitions $S = \{s_1, s_2, \dots, s_n\}$ and $T = \{t_1, t_2, \dots, t_n\}$ of equal size $|S| = |T| = n$, representing each vertex v_i twice: as a source s_i and as a target t_i . E' contains an undirected edge $\{s_i, t_j\}$ iff E contains a directed edge (v_i, v_j) . Furthermore, we represent f^+ and f^- together by a function $f^\pm : S + T \rightarrow \mathbb{R}$:*

$$f^\pm(s_i) = f^+(v_i)$$

$$f^\pm(t_i) = f^-(v_i)$$

The transformation to an uncoiled graph is just a matter of representation; from U and f^\pm , the original G and f^-, f^+ can easily be recovered. An example directed graph G and its uncoiled graph U in two different arrangements are shown in Fig. 2. Every vertex v_i in G corresponds to two vertices s_i and t_i in U (in Fig. 2b, these are kept close to the spot of v_i). Every edge in G corresponds to an edge in U ; if it leaves from v_i and enters v_j , its corresponding edge in U is incident to s_i and t_j .

Fig. 2a also shows partial vertex sum requirements for G : $f^+(v_1) = 5$ and $f^-(v_1) = 3$, and a partial weight distribution that matches these requirements: $f(v_1, v_1) = 3$ and $f(v_1, v_2) = 2$. In fact, this f has been deduced from f^- and f^+ : because v_1 has only one incoming edge, we can solve $f(v_1, v_1) = f^-(v_1) = 3$. With this information, we can also solve $f(v_1, v_2) = f^+(v_1) - f(v_1, v_1) = 5 - 3 = 2$. This illustrates the basic principle of how Algorithm 1 works.

For graph U , these same requirements and distribution are represented by f^\pm and f' , respectively (see Fig. 2b, 2c). The assertion that f' matches f^\pm is

$$\forall v \in S + T. \quad f^\pm(v) = \sum_{w|\{v,w\} \in E'} f'\{v,w\}$$

The algorithm works on this new representation: it solves f' from f^\pm . Afterwards, the solution f' is translated to the corresponding f . We now state the sufficient condition to find this solution: U should not contain a cycle.

Lemma 1. *A cycle in U represents a direction-alternating cycle in G and vice versa. A direction-alternating cycle is a sequence of an even number of distinct directed edges $(e_1, e_2, \dots, e_{2m})$ in which:*

- e_1 and e_{2m} have a common source
- e_i and e_{i+1} have a common target, for odd i
- e_i and e_{i+1} have a common source, for even i (smaller than $2m$)

Theorem 2. *For each directed graph G without direction-alternating cycles and weight-sum functions f^+ and f^- , if there exists a matching weight distribution f , it is unique. Algorithm 1 decides whether it exists; if so, it produces the solution.*

Proof. The algorithm works on the uncoiled graph U , which contains no cycles because of Lemma 1; hence, it is a forest. For each component tree, we pick an arbitrary element as root and recurse over the subtrees. The proof is by induction over the tree structure; the induction hypothesis is that after a call $\text{SolveSubtree}(\text{root}, \text{maybeParent})$, the unique matching distribution on all the edges in the subtree rooted in root has been recorded in f' . To satisfy the hypothesis for the next level, we consider the f^\pm requirements for the roots of the subtrees. By the induction hypothesis, these all have one unknown term, corresponding to the edge to their parent: thus, we find a unique solution for each edge at the current level. \square

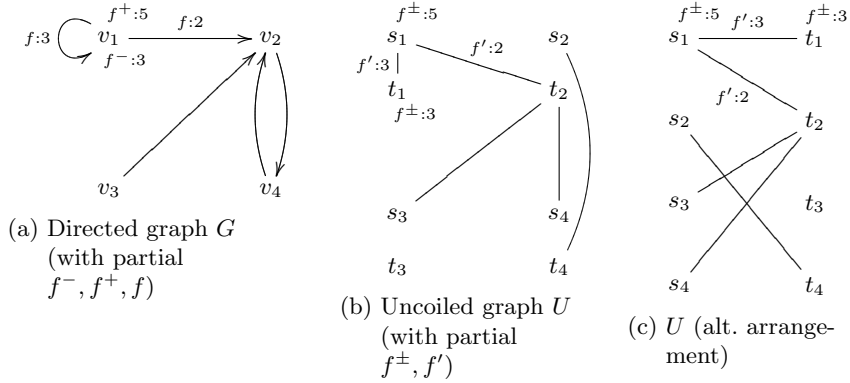


Fig. 2: Uncoiling

```

Input: directed graph  $G = (V, E)$  and weight sums  $f^+, f^-$ 
Output: unique weight distribution  $f$  that matches the sums
 $(V', E') \leftarrow$  uncoiled graph of  $G$  (as in Def. 1)
 $f^\pm \leftarrow$  projection of  $f^+, f^-$  on  $V'$  (as in Def. 1)
 $f' \leftarrow$  the empty (partial) function
 $visited \leftarrow \emptyset$ 

while  $visited \neq V'$  do
   $root \leftarrow$  an arbitrary element of  $(V' - visited)$ 
  SolveSubtree( $root, \emptyset$ )
  if  $f^\pm(root) \neq \sum_w f'\{root, w\}$  then error 'no distribution exists'
end
foreach  $(v_i, v_j) \in E$  do  $f(v_i, v_j) \leftarrow f'\{s_i, t_j\}$ 
procedure SolveSubtree( $root, maybeparent$ )
  //  $maybeparent$  records the node we came from, to prevent going back
  if  $root \in visited$  then error 'cycle detected'
   $visited \leftarrow visited \cup \{root\}$ 
  foreach  $v \in (V' - maybeparent)$  such that  $\{root, v\} \in E'$  do
    SolveSubtree( $v, \{root\}$ )
     $f'\{root, v\} \leftarrow f^\pm(v) - \sum_w f'\{v, w\}$ 
  end
end

```

Algorithm 1: Finding the unique weight distribution

Remark. The algorithm contains two additional checks:

- When the root of a component is reached, the f^\pm equation for this root is checked. If it holds, the *existence* of a matching distribution f is established (for this component).
- When visiting a node, it is checked whether it was visited before. If it was, U contains a cycle. As we will show next, this means that no *unique* matching distribution exists.

Theorem 3. *A directed graph G with a direction-alternating cycle has no unique weight distribution matching a given f^+, f^- .*

Proof. Given such a cycle $(e_1, e_2, \dots, e_{2m})$ and a matching weight distribution f , we construct another matching weight distribution g :

$$\begin{aligned}
 g(e_i) &= f(e_i) + c, & \text{for all odd } i \\
 g(e_i) &= f(e_i) - c, & \text{for all even } i \\
 g(e) &= f(e), & \text{for all other edges } e
 \end{aligned}$$

□

5 Finishing the Transition Model Construction

In Sect. 3, we ended with a set of equations to solve, namely

$$\begin{aligned} \sum_{h \llbracket h \rrbracket \neq \llbracket i \rrbracket} F(h, i) &= \pi_G(i) - \sum_{h \llbracket h \rrbracket = \llbracket i \rrbracket} F(h, i) \\ \sum_{j \llbracket j \rrbracket \neq \llbracket i \rrbracket} F(i, j) &= \pi_G(i) - \sum_{j \llbracket j \rrbracket = \llbracket i \rrbracket} F(i, j) \end{aligned}$$

for all border states i . To solve these, we use Algorithm 1, with (V, E) the inter-component transition graph: vertices V are the border states, and edges E are the inter-component transitions. The unknown inter-component transition probabilities that we want to solve correspond to the edge weights in f , and the vertex sum requirements correspond to the right-hand sides of the above equations:

$$\begin{aligned} f^-(i) &= \pi_G(i) - \sum_{h \llbracket h \rrbracket = \llbracket i \rrbracket} F(h, i) \\ f^+(i) &= \pi_G(i) - \sum_{j \llbracket j \rrbracket = \llbracket i \rrbracket} F(i, j) \end{aligned}$$

The weight distribution f that the algorithm yields gives us the unknown F values. So, we now know $F(i, j)$ for all (i, j) :

$$\begin{aligned} F(i, j) &= \pi_{\llbracket i \rrbracket}(i) \cdot \llbracket i \rrbracket(i, j) \text{ for all intra-component } (i, j) \text{ transitions} \\ F(i, j) &= f(i, j) \text{ for } (i, j) \text{ in the inter-component transition graph} \\ F(i, j) &= 0 \text{ for other } (i, j) \text{ (for which } G(i, j) = 0) \end{aligned}$$

We arrive at the desired matrix G of conditional probabilities by normalizing the rows:

$$G(i, j) = \frac{F(i, j)}{\sum_x F(i, x)}$$

6 Conclusion and Future Work

We have presented a technique to compose Markov models of disjunct parts of a state space into a large Markov model of the entire state space. We review the conditions under which we can apply this technique:

- The inter-component transition graph should be known, and should not contain any direction-alternating cycles.
- The Markov chain G should be irreducible and ergodic, in order to calculate the stationary distribution π_G . We refer to [2] for the definition of these terms; for finite chains, it suffices that all states are accessible from each other (i.e. the transition graph is strongly connected) and that all states are *aperiodic*: the possible numbers of steps in which you can return to a given state should not be only multiples of $n > 1$.

An example class of inter-component transition graphs satisfying these conditions is formed by those with symmetric edges (only two-way transitions, and possibly self-transitions), that forms a *tree* when the two-way connections are represented by an undirected edge and the self-transitions are left out.

In this article, we have only considered the situation where the C_i models are perfect pseudo-aggregations of one model G . In practice, this will probably never be the case. Even when the observation sequences are generated by a perfect Markov model, they would have to be infinitely long to guarantee this. The consequence using imperfect pseudo-aggregations is that the stationary distributions π_{C_i} will not perfectly agree with another, and π_G is only *approximated*. We leave it to future research to determine when an acceptable approximation can be reached. A second open question is how to deal with inter-model transition graphs which *do* contain some direction-alternating cycles; perhaps some additional information could be used to determine the best solution.

Acknowledgements

This research is funded by NWO (Nederlandse Organisatie voor Wetenschappelijk Onderzoek; Netherlands Organisation for Scientific Research), under project 639.022.403. The authors would like to thank Richard Boucherie for his comments on the article.

References

1. Rubino, G., Sericola, B.: Sojourn times in finite Markov processes. *Journal of Applied Probability* **26**(4) (December 1989) 744–756
2. Ross, S.M.: *Introduction to Probability Models*. 8th edn. Academic Press (2003)