An Expectation-Maximization Algorithm to Compute a Stochastic Factorization From Data

André M. S. Barreto and Rafael L. Beirigo
Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil
{amsb,rafaelb}@lncc.br

Joelle Pineau and Doina Precup
McGill University
Montreal, QC, Canada
{jpineau,dprecup}@cs.mcgill.ca

Abstract

When a transition probability matrix is represented as the product of two stochastic matrices, swapping
the factors of the multiplication yields another transition matrix that retains some fundamental char-
acteristics of the original. Since the new matrix can be much smaller than its precursor, replacing
the former for the latter can lead to significant savings in terms of computational effort. This strat-
ey, dubbed the “stochastic-factorization trick,” can be used to compute the stationary distribution of a
Markov chain, to determine the fundamental matrix of an absorbing chain, and to compute a de-
cision policy via dynamic programming or reinforcement learning. In this paper we show that the
stochastic-factorization trick can also provide benefits in terms of the number of samples needed to
estimate a transition matrix. We introduce a probabilistic interpretation of a stochastic factorization
and build on the resulting model to develop an algorithm to compute the factorization directly from
data. If the transition matrix can be well approximated by a low-order stochastic factorization, es-
timating its factors instead of the original matrix reduces significantly the number of parameters to
be estimated. Thus, when compared to estimating the transition matrix directly via maximum likeli-
hood, the proposed method is able to compute approximations of roughly the same quality using less
data. We illustrate the effectiveness of the proposed algorithm by using it to help a reinforcement learn-
ing agent learn how to play the game of blackjack.

1 Introduction

Results from the supervised learning literature indicate that
general efficient learning is only possible if one exploits some
kind of regularity in the problem [Györfi et al., 2002]. When
learning a model that describes a stochastic process, one can
exploit, for example: a sparse or low-rank transition matrix, a
smooth transition kernel, or the fact that the data lies close to a
lower dimensional manifold [Farahmand, 2011]. Such regu-
larities allow estimating parameters with a reasonable amount
of data [Györfi et al., 2002].

One type of structural regularity that has recently received
attention is stochastic factorization [Barreto and Fragoso,
2011]. In this case, a transition probability matrix of size
$n \times n$ is represented as the product of two stochastic matrices,
of sizes $n \times m$ and $m \times n$, where $m < n$. Interestingly, swapping
the factors of the multiplication yields another stochastic
matrix, of size $m \times m$, which retains some fundamental
characteristics of its precursor [Barreto and Fragoso, 2011].
This makes it possible to use the new matrix instead of the
original—which can lead to significant savings in terms of
computational effort if $m$ is considerably smaller than $n$.
This strategy, dubbed the “stochastic-factorization trick,” has
been used to compute the stationary distribution of a Markov
chain [Barreto and Fragoso, 2011], to determine the funda-
mental matrix of an absorbing chain [Barreto and Fragoso,
2011], and to compute decision policies in large problems
through dynamic programming [Barreto et al., 2014] and rein-
forcement learning [Barreto et al., 2011].

So far, the stochastic factorization problem has appeared
in the literature with a strict “algebraic” formulation: given
a transition matrix, find two lower-order stochastic matrices
whose multiplication approximates the original matrix as well
as possible [Barreto and Fragoso, 2011; Barreto et al., 2014].
In this paper we are interested in computing a stochastic fac-
torization when the transition matrix is not known, and we
only have access to transitions sampled from it. One possi-
ble approach in this case is to use the samples to compute an
estimate of the transition matrix, which takes us back to the
scenario described above. However, this approach has the
disadvantage that many entries may need to be estimated, per-
haps from not much data. Instead, we would like to estimate
the factors directly from data, because this entails potentially
many fewer parameters. We introduce a probabilistic inter-
pretation of a stochastic factorization and build on the result-
ing model to derive an expectation-maximization algorithm
that does just that.

Since the order of the factorization (i.e., the value of $m$)
is a parameter of the proposed algorithm, by changing it one
can control the complexity of the resulting model, and thus
the trade off between the approximation and estimation er-
ors [Györfi et al., 2002]. This means that, if the transition
matrix can be well approximated by a low-order stochastic
factorization, the stochastic-factorization trick may provide
benefits not only from a computational point of view but also
in terms of the number of sample transitions needed in the approximation. We empirically show that, when compared to estimating the transition matrix directly via maximum likelihood (that is, by “counting” events), the proposed method is able to compute approximations of roughly the same quality using less data. In order to illustrate the practical utility of our algorithm, we use it to learn a model of the game of blackjack.

2 Background and Notation
This section introduces the notation adopted and briefly reviews some concepts that will be used throughout the paper.

Random variables are represented by capital letters; we use the notation $A_{1:t}$ to refer to a sequence of variables $A_1, A_2, \ldots, A_t$. Scalar variables are represented by small letters; boldface capital letters and boldface small letters are used to denote matrices and vectors, respectively. We will need the following definitions:

Definition 1 A matrix $P \in \mathbb{R}^{n \times q}$ is called stochastic if and only if $p_{ij} \geq 0$ for all $i, j$ and $\sum_{j=1}^{q} p_{ij} = 1$ for all $i$. A square stochastic matrix is called a transition matrix.

Definition 2 Given a stochastic matrix $P \in \mathbb{R}^{n \times q}$, the relation $P = DK$ is called a stochastic factorization of $P$ if $D \in \mathbb{R}^{n \times m}$ and $K \in \mathbb{R}^{m \times q}$ are also stochastic matrices. The integer $m > 0$ is the order of the factorization.

Definition 3 The stochastic rank of a stochastic matrix $P \in \mathbb{R}^{n \times q}$, denoted by $\text{srk}(P)$, is the smallest possible order of the stochastic factorization $P = DK$.

Given a stochastic factorization of a transition matrix, $P = DK$, swapping the factors of the factorization yields another transition matrix $\tilde{P} = KD$, potentially much smaller than the original, which retains the basic topology of $P$—that is, the number of recurrent classes and their respective reducibilities and periodicities (see Barreto and Fragoso’s [2011] article for formal definitions). Since $\tilde{P}$ can be considerably smaller than $P$, the idea of replacing the latter with the former comes almost inevitably: this is the “stochastic-factorization trick.”

3 Stochastic Factorization as a Probabilistic Model
So far, an algebraic view of the stochastic factorization has prevailed in the literature, with $P$, $D$, and $K$ treated simply as matrices with a particular structure. In this paper we introduce a probabilistic interpretation of a stochastic factorization. In particular, we see the factorization as a Markov model that represents the joint distribution of a sequence of random variables.

3.1 Stochastic Factorization Model
Let $S = \{1, 2, \ldots, n\}$, $H = \{1, 2, \ldots, m\}$, and $A = \{1, 2, \ldots, n_0\}$. Consider the stochastic process $(S_1, A_1, H_1, S_2, A_2, H_2, \ldots)$, where $S_t \in S$ are observable states, $A_t \in A$ are observable actions (or decisions), and $H_t \in H$ are hidden states. Our probabilistic model builds on the following Markov assumptions:

(i) $\Pr(S_t | H_{t-1}, A_{t-1}, S_{t-1}, \ldots, S_1) = \Pr(S_t | H_{t-1}, A_{t-1})$;

(ii) $\Pr(H_t | A_t, S_t, H_{t-1}, \ldots, S_1) = \Pr(H_t | A_t, S_t)$;

(iii) $\Pr(A_t | S_t, H_{t-1}, A_{t-1}, \ldots, S_1) = \Pr(A_t | S_t)$.

Given $\tau > 0$, it is easy to compute the probability of any sequence of length $\tau$ using (i), (ii), and (iii):

$$
\Pr(S_1, A_1, H_1, \ldots, S_{\tau-1}, A_{\tau-1}, H_{\tau-1}, S_{\tau}) = \Pr(S_1) \prod_{t=2}^{\tau} \Pr(S_t | H_{t-1}, A_{t-1}, S_{t-1}) \times \Pr(A_{t-1} | S_{t-1}, H_{t-2}, \ldots, S_1) \Pr(H_{t-1} | A_{t-1}, S_{t-1}, \ldots, S_1) = \Pr(S_1) \prod_{t=2}^{\tau} \Pr(S_t | H_{t-1}, A_{t-1}) \Pr(A_{t-1} | S_{t-1}) \times \Pr(H_{t-1} | S_{t-1}, A_{t-1}).
$$

Thus, in order to define our model we only need to represent $\Pr(S_1)$, $\Pr(S_{t+1} | H_t, A_t)$, $\Pr(A_t | S_t)$, and $\Pr(H_t | S_t, A_t)$. This leads to the following definition:

Definition 4 A stochastic factorization model (SFM) is a tuple $F = (D^a, K^a, \Pi, \mu)$, where $D^a \in \mathbb{R}^{n \times m}$, $K^a \in \mathbb{R}^{m \times q}$, and $\mu \in \mathbb{R}^{n \times n_0}$ are stochastic matrices, one for each $a \in A$. $\Pi \in \mathbb{R}^{n \times n_0}$ is a stochastic matrix, and $\mu \in \mathbb{R}^n$ is a distribution. Based on Assumptions (i), (ii), and (iii), the model represents the stochastic process $(S_1, A_1, H_1, S_2, A_2, H_2, \ldots)$ as

$$
\Pr(S_t = j | S_t = i, A_t = a) = \prod_{t=1}^{\tau} \Pr(H_t = i | S_t = i, A_t = a), \quad \Pr(A_t = a | S_t = i) = \mu_i, \quad \Pr(S_t = i).
$$

Note that a SFM with $n_0 = 1$ is an uncontrolled system. In order to provide some intuition about our model, in Figure 1 we compare it with another uncontrolled system, the well known hidden Markov model (HMM) [Rabiner, 1989]. As shown, the main difference between the two models is the fact that in a SFM knowledge of the current observable state is sufficient to define the distribution of the next hidden state.

3.2 Computing Probabilities
Recall that in a SFM only the variables $S_t$ and $A_t$ are observable. Let $z_{1:t} = (s_1, a_1, s_2, \ldots, a_t, s_t)$. In order to do inference with our model, we need to be able to compute the probability of a given sequence happening, that is, $\Pr(Z_{1:t} = z_{1:t}, H_t = i(\lambda))$, where $\lambda \equiv (D^a, K^a, \Pi, \mu)$. In this section we discuss how to compute this quantity. We base our strategy on the forward-backward procedure to compute probabilities in an HMM [Baum, 1972; Rabiner, 1989].

We start by noting that, if we can compute $\Pr(Z_{1:t} = z_{1:t}, H_t = i(\lambda))$, then we can compute the desired quantity by marginalizing over $H_t$. Given $z_{1:t}$, we define the “forward variable” $\alpha_t(i)$ as:

$$
\alpha_t(i) = \Pr(Z_{1:t} = z_{1:t}, A_t = a_t, H_t = i(\lambda)).
$$

In Table 1 we show that, given a SFM, we can compute $\alpha_t(i)$ for any $i$ and any $t$ using the following recursive formulae:

$$
\alpha_t(i) = \sum_{a_{t+1}} \alpha_t(t+1) = \sum_{a_{t+1}} \alpha_t(i) K^a_{i a_{t+1}} \prod_{t+1}^{\tau} \Pr(H_t = i | S_t = i, A_t = a_t),
$$

where $a_{t+1} = \arg \max_{a_t} \alpha_t(i) K^a_{i a_{t+1}} \prod_{t+1}^{\tau} \Pr(H_t = i | S_t = i, A_t = a_t)$.

---

Figure 1: Graphical models of HMMs and SFMs.
We now define the “backward variable” $\beta_i(t)$ as:

$$\beta_i(t) = \Pr(Z_{t+\tau} = z_{t+1}, H_t = i | A_t = a_t, \lambda).$$

As with $\alpha_i(t)$, given a SFM, the variable $\beta_i(t)$ can be computed recursively, in the following way (Table 1):

$$\beta_i(t - 1) = k_{a_{t+1}i}^{w_{t+1}-1} \sum_j \beta_j(t) d_{ij}^a \pi_{a_0a_1} k_i^{a_{t+1}}.$$  

Now that we are able to compute $\alpha_i(t)$ and $\beta_i(t)$ for any $i$ and any $t$, we can compute the desired probability by simply multiplying these variables, that is,

$$\alpha_i(t) \beta_i(t) = \Pr(Z_{t+\tau} = z_{t+1}, H_t = i | \lambda).$$

The equality above is also justified in Table 1.

**Scaling**

Using the variables $\alpha_i(t)$ and $\beta_i(t)$ we can compute $\Pr(Z_{t+\tau}, H_t = i | \lambda)$, which in turn allows us to compute the probability of any sequence being generated by a SFM. However, since the variables $\alpha_i(t)$ get very small as $t \to \infty$ (and equivalently for $\beta_i(t)$ as $t \to 0$), in practice the recursive equations presented above may lead to numerical instability. We present a simple workaround, based on Rabiner’s [1989] solution for a similar problem, which consists in rescaling the variables $\alpha_i(t)$ and $\beta_i(t)$ after each recursion.

Let $w_t = \sum \alpha_i(t)$. Starting with $\alpha_i(1) = \alpha_i(1)/w_1$, we will apply the following recursion, for $t \geq 2$:

$$\alpha'_i(t) = \sum_j \alpha_i(t - 1) k_{a_{t+1}j}^{w_{t+1}-1} \pi_{a_0a_1} d_{ij}^a,$$

$$\beta'_i(t) = \frac{\alpha'_i(t)}{w'_t},$$

where $w'_t = \sum \alpha'_i(t)$. Using induction, it is easy to show that

$$\hat{\alpha}_i(t) = \prod_{j=1}^t \frac{1}{w_j} \alpha_i(t) = \frac{1}{w_t} \alpha_i(t),$$

and thus the variables $\hat{\alpha}_i(t)$ define a distribution (and are in general representable with standard machine precision).

From Table 1 we know that

$$\Pr(Z_{t+\tau} | \lambda) = \sum_{i=1}^m \alpha_i(t - 1) \beta_i(t - 1) = \prod_{j=1}^{	au-1} w'_j \sum_{i=1}^m \hat{\alpha}_i(t - 1) k_{a_{t+1}i}^{w_{t+1}-1},$$

which means that, as long as the variables $w'_t$ are available, we can use the scaled forward variables to compute the probability of any sequence in a SFM.

Although we do not need the backward variables to compute $\Pr(Z_{t+\tau} | \lambda)$, it will be convenient to also define their scaled version. Starting with $\hat{\beta}_i(t - 1) = \beta_i(t - 1)$ and $\hat{\beta}_i(t - 2) = \beta_i(t - 2)/w_{t-1}$, we will apply the following recursion, for $t < \tau - 2$:

$$\hat{\beta}_i(t - 1) = \sum_j \hat{\beta}_j(t) d_{ij}^a \pi_{a_0a_1} k_i^{a_{t+1}}$$

$$\hat{\beta}_i(t - 1) = \frac{\hat{\beta}_i(t - 1)}{w'_t}. $$

Note that, since the magnitudes of $\beta'_i(t - 1)$ and $\alpha'_i(t)$ are comparable, the computation of $\hat{\beta}_i(t)$ will also be numeri-
cally stable. One can show that
\[ \hat{\beta}_t(t) = \prod_{j=t+1}^{\tau} \frac{1}{w_j^t} \hat{\alpha}_j(t). \]  
(9)

From (6) and (9) we know that \( \hat{\alpha}_t(t) \hat{\beta}_t(t) = \prod_{j=t+1}^{\tau} (w_j^t)^{-1} \alpha_j(t) \beta_j(t) \), which, combined with (4), implies that
\[ \frac{\hat{\alpha}_t(t) \hat{\beta}_t(t)}{\sum_i \hat{\alpha}_i(t) \hat{\beta}_i(t)} = \frac{\alpha_j(t) \beta_j(t)}{\sum_i \alpha_i(t) \beta_i(t)} = \Pr(H_t = i | Z_{1: \tau}, \lambda). \]  
(10)

Equation (10) will be useful for learning a SFM from data, as we discuss next.

4 Expectation-Maximization Algorithm

One of the motivations for introducing the SFM is to provide a framework for the application of the stochastic-factorization trick directly from data. The idea is as follows: given a sequence of observation-action pairs coming from a set of \( n \) transition matrices \( \Pi^\alpha \), instead of estimating the matrices \( \Pi^\alpha \) directly, we estimate a SFM \( F = (D^\alpha, K^\alpha, \Pi, \mu) \). By doing so, we automatically have approximations \( D^\alpha K^\alpha \approx \Pi^\alpha \).

The strategy above has two benefits. First, by estimating \( D^\alpha \) and \( K^\alpha \) instead of \( \Pi^\alpha \), we reduce the number of parameters we are estimating from \( n^2 \) to \( 2nm \). Second, once we have the SFM, we do not need to actually compute the multiplications \( D^\alpha K^\alpha \); instead, we compute \( \bar{P}^\alpha = K^\alpha D^\alpha \) and use this smaller matrix in place of \( \Pi^\alpha \).

In this section we discuss how to compute a SFM directly from data. Since SFMs are a latent-variable model, we resort to the well known expectation-maximization (EM) algorithm [Dempster et al., 1977]. We assume the reader is familiar with the basic principles of EM; for good reviews of the subject see for example Bishop’s [2006] book or Bilmes’s [1998] tutorial.

4.1 Derivation

Given a sequence of observation-action pairs, \( z_{1: \tau} \), our goal is to compute a SFM that maximizes the likelihood of the data, \( \mathcal{L}(\lambda | z_{1: \tau}) = \Pr(z_{1: \tau} | \lambda) \). As well known, the EM algorithm does this through the function
\[ Q(\lambda, \lambda') = \sum_{h, z} \log \Pr(h_{1: \tau}, Z_{1: \tau} | \lambda) \Pr(h_{1: \tau} | Z_{1: \tau}, \lambda'). \]  
(11)

In the “E” step we use the current parameter values \( \lambda' \) to compute the expectation above; in the “M” step we maximize the expectation we computed with respect to \( \lambda \). In this section we will show how to derive update equations to compute a SFM based on the above framework. In order to improve readability we will drop the subscript \( 1: \tau \) throughout the section, that is, we will use \( h, Z \), and \( z \) to refer to \( h_{1: \tau}, Z_{1: \tau}, \) and \( z_{1: \tau} \).

As shown in (1), in the case of a SFM we have
\[ \Pr(h, Z | \lambda) = \mu_{z_i} \prod_{t=1}^{\tau-1} \pi_{s_{at}} d_{s_{at} h_{t+1}}^{\alpha_{s_{at}}} k^{\alpha_{s_{at}}}_{s_{at} h_{t+1}}. \]  
(12)

Substituting (12) in (11), the \( Q \) function becomes
\[ Q(\lambda, \lambda') = \sum_{h \in \mathcal{H}} \log \mu_z \Pr(h | Z, \lambda') + \sum_{h \in \mathcal{H}} \sum_{t=1}^{\tau-1} \log \pi_{s_{at}} \Pr(h | Z, \lambda') + \sum_{h \in \mathcal{H}} \sum_{t=1}^{\tau-1} \log d_{s_{at} h_{t}}^{\alpha_{s_{at}}} \Pr(h | Z, \lambda') + \Delta_D \]  
\[ \Delta_K \]  
\[ \Delta_K = \sum_{h \in \mathcal{H}} \sum_{t=1}^{\tau-1} \log k^{\alpha}_{s_{at} h_{t+1}} \Pr(h | Z, \lambda'). \]  
(12)

(recall that here \( \mathcal{H} \) represents the space of all possible sequences \( h \) of length \( \tau \)). We will focus on the third term of the equation above, tagged as “\( \Delta_D \),” to show how to derive update equations for the elements of the matrices \( D^\alpha \); the update equations for the remaining parameters of a SFM are obtained in an analogous way. Our derivation closely follows that of Bilmes [1998].

Given \( z \), suppose that \( s_t = i \) and \( a_t = u \). Then, for each value of \( j \in \{1, 2, ..., m\} \), the corresponding term \( \log d_{ij}^u \) appearing in \( \Delta_D \) will multiply \( \Pr(h | Z, \lambda') \) if and only if the \( j \)th element in \( h \) is \( j \)—that is, \( d_{ij}^u \) will multiply \( \sum_{h \in \mathcal{H}} \Pr(h | Z, \lambda') \{ h_t = j \} = \Pr(h_t = j | Z, \lambda') \) (here \( 1 \{ \cdot \} \) is the indicator function). Thus, we can rewrite \( \Delta_D \) as:
\[ \Delta_D = \sum_{u=1}^{m} \sum_{j=1}^{m} \log d_{ij}^u \sum_{t=1}^{\tau-1} \Pr(h_t = j | Z, \lambda') \{ s_t = i, a_t = u \}. \]  
(13)

Now, if we make \( \partial \Delta_D / \partial d_{ij}^u = 0 \), we have a necessary condition for \( d_{ij}^u \) to be an extreme point of \( Q \). Since
\[ \frac{\partial \Delta_D}{\partial d_{ij}^u} = \frac{1}{d_{ij}^u} \sum_{t=1}^{\tau-1} \Pr(h_t = j | Z, \lambda') \{ s_t = i, a_t = u \} - \kappa_i^u, \]  
we know that
\[ d_{ij}^u = \left( \frac{1}{\Delta_D} \sum_{t=1}^{\tau-1} \Pr(h_t = j | Z, \lambda') \{ s_t = i, a_t = u \} \right) / \kappa_i^u. \]  
(13)

If we sum (13) over \( j \), using the fact that \( \sum_j d_{ij}^u = 1 \), we obtain \( \kappa_i^u = \sum_{t=1}^{\tau-1} \{ s_t = i, a_t = u \} \). Substituting \( \kappa_i^u \) back in (13), we have the following update rule for \( d_{ij}^u \):
\[ d_{ij}^u = \left( \frac{\sum_{t=1}^{\tau-1} \Pr(h_t = j | Z, \lambda') \{ s_t = i, a_t = u \}}{\sum_{t=1}^{\tau-1} \{ s_t = i, a_t = u \}} \right). \]  
(14)

If we follow the same steps as above replacing \( \Delta_D \) with \( \Delta_K \) we arrive at the following update equation for \( k^{\alpha}_{ij} \):
\[ k^{\alpha}_{ij} = \left( \frac{\sum_{t=1}^{\tau-1} \Pr(h_t = i | Z, \lambda') \{ s_{at+1} = j, a_t = u \}}{\sum_{t=1}^{\tau-1} \Pr(h_t = i | Z, \lambda') \{ a_t = u \}} \right). \]  
(15)
As discussed in Section 3.2, we can compute \( \Pr(H_t = i | Z, \lambda) \) using (10).

Update equations (14) and (15) make intuitive sense: loosely speaking, they are the equations for computing the relative frequency of events with the occurrence of \( H_t = i \) replaced by its probability. Based on this insight, and following Rabiner’s [1989] reasoning, we can extend our algorithm to the case in which we have \( c \) sequences of observations and actions, \( z'_{1:τ} = \{ s'_{1}, a'_1, \ldots, a'_{τ-1}, s'_{τ} \} \), with \( i = 1, 2, \ldots, c \).

In this case we have the following update rules:

\[
d_{ij}^t = \sum_{t=1}^{τ} \sum_{j=1}^{m} \Pr(H_t = j | Z = z', \lambda) 1 \{ s'_t = i, a'_t = u \} / \sum_{i=1}^{c} \sum_{j=1}^{m} \Pr(H_t = j | Z = z', \lambda) 1 \{ s'_t = i, a'_t = u \}
\]

\[
b_{ij}^t = \frac{\sum_{t=1}^{τ} \sum_{j=1}^{m} \Pr(H_t = j | Z = z', \lambda) 1 \{ s'_t+1 = j, a'_t = u \} }{ \sum_{i=1}^{c} \sum_{j=1}^{m} \Pr(H_t = j | Z = z', \lambda) 1 \{ s'_t = i, a'_t = u \} }.
\]

(16)

(17)

In principle, we could follow the steps above to derive update equations for \( μ_i \) and \( π_{ti} \). Note though that, since these parameters are based on observable quantities, the associated update equations will be simply the relative frequency of the corresponding events, that is:

\[
μ_i = \frac{\sum_{t=1}^{τ} 1 \{ s'_t = i \} }{c}, \quad π_{ti} = \frac{\sum_{t=1}^{τ} \sum_{j=1}^{m} 1 \{ s'_t = i, a'_t = u \} }{ \sum_{i=1}^{c} \sum_{j=1}^{m} \sum_{j=1}^{m} 1 \{ s'_t = i \} }.
\]

(18)

Since the quantities appearing in (18) do not depend on the parameters of the SFM, the estimates of \( μ \) and \( Π \) will be fixed throughout the EM iterations. Note that in scenarios where we have control over the collection of data we generally know \( μ \) and \( Π \), which means that these parameters do not have to be estimated at all.

To conclude, we note that for both practical and theoretical reasons the stochastic-factorization trick is sometimes applied with a single \( D \) or a single \( K \) (see Barreto et al., 2011 and Barreto et al., 2014, respectively). One can easily specialize the proposed method to these particular cases. Having a single \( K \), for example, corresponds to replacing Assumption (ii) with \( \Pr(H_t | A_t, S_t, H_{t-1}, \ldots, S_1) = \Pr(H_t | S_t) \). The resulting update equation will be expression (17) without the restriction \( a'_t = u \).

4.2 Algorithm

Algorithm 1 shows a step by step description of the proposed method, which we call ESMF. The pseudo-code uses two matrices to represent the forward and backward variables: \( α \in \mathbb{R}^{τ-1 \times m} \), where \( α_{ti} = α_i(t) \), and \( β \in \mathbb{R}^{τ-1 \times m} \), where \( β_{ti} = β_i(t) \). For each sequence \( z'_{1:τ}, \) ESMF first computes \( α \) using (5) and then computes \( β \) using (8). Then, \( α \) and \( β \) are multiplied element-wise, giving rise to matrix \( C \in \mathbb{R}^{τ-1 \times m} \). After the rows of \( C \) have been normalized, as shown in line 39 of Algorithm 1, we have \( c_{ij} = \Pr(H_t = j | Z_{1:τ} = z', \lambda) \) (see (10)). These values are then used to compute the numerator of update equations (16) and (17), which are accumulated in the auxiliary matrices \( \hat{D}^a_{\tau} = \mathbb{R}^{m \times n} \) and \( \hat{K}^a = \mathbb{R}^{m \times n} \). After all the sequences have been processed, the rows of \( \hat{D}^a \) and \( \hat{K}^a \) are normalized (lines 51 and 52 of Algorithm 1), which corresponds to completing the application of (16) and (17) to obtain the new estimates of \( D^a \) and \( K^a \), with \( a \in A \).
and $K^a$. The process continues until the decrease on the data log likelihood falls below a given threshold.

EMSF is an expectation-maximization algorithm, and thus it will only converge to a local maximum of $L$. The solution the algorithm converges to is determined by the initial values of $D^a$ and $K^a$. Besides its initialization, the approximation computed by EMSF depends only on one parameter, $m$. The parameter $m$ controls the complexity of the SFM computed by EMSF, and thus it can be seen as a "regularizer" of the approximations $D^aK^a \approx P^a$ (see discussion in Section 5) [Györfi et al., 2002; Hastie et al., 2002].

As for computational complexity, each iteration of EMSF is $O(\tau m)$, where $\tau = \sum_i \bar{\eta}_i$. Thus, the cost of the algorithm is not a direct function of $n$, the size of the matrices $P^a$ being estimated. Of course, $n$ will play a role in the definition of an appropriate value for $\tau$, since EMSF estimates $2a\alpha nm$ parameters.

5 Experiments

In this section we present computational experiments to illustrate the properties and usefulness of our algorithm. The first experiment is a "proof of concept": we generated transition matrices $P \in \mathbb{R}^{100 \times 100}$, with $\text{srk}(P) = 20$, and tried to recover them with EMSF using different values for $m$. We then compared EMSF's results with those obtained by directly estimating $P$ via maximum likelihood (referred to in the plots as CNT, for "counting"). The results are shown in Figure 2.

Since EMSF estimates fewer parameters than CNT, its approximation error decreases faster with the number of sample transitions. However, since the EM algorithm only converges to a local maximum of $L$, EMSF's approximation error stagnates at a positive value, even when $m = \text{srk}(P)$, while CNT's approximation keeps improving as $\tau \to \infty$.

These results are exactly as expected. As CNT is statistically consistent, we know that its approximation error will converge to zero, but this comes at the price of higher estimation errors at lower values of $\tau$. EMSF reduces the estimation error by enforcing a specific structure on $P$—in terms of the classical bias-variance analysis, EMSF increases bias and decreases variance when $m < n$ [Hastie et al., 2002].

This view of EMSF also helps to explain the impact of the parameter $m$ over its performance. Since $m$ directly controls the trade off between bias and variance, when computing a stochastic factorization from data the "optimal" value for this parameter depends not only on $\text{srk}(P)$, but also on the amount of data available. In general, as the number of transitions increases so does the best value for $m$ (this is, again, exactly the trend observed in Figure 2). The definition of an appropriate value for $m$ is thus an instance of the well known model selection problem; one can address this problem using cross validation [Hastie et al., 2002].

It is clear then that EMSF can be seen as a strategy to regularize the approximation of $P$ in order to reduce the number of transitions needed for learning. An interesting question is whether the structure imposed on $P$ by our algorithm really arises in practice. In order to answer this question, we used EMSF to compute a model of the game of blackjack.

The game of blackjack was implemented exactly as described in Section 5.1 of Sutton and Barto’s [1998] book. The experiments were carried out as follows. First, an exploration policy that selects actions uniformly at random played a number of hands of blackjack. The resulting sample transitions were then used by CNT to compute estimates of both the transition matrices $P^a$ and the expected-reward vectors $r^a$ of the Markov decision process (MDP) describing the game. Finally, policy iteration (PI) was applied to the MDP to compute a policy, which was evaluated on $10^6$ hands of blackjack.

In order to evaluate EMSF we used an algorithm introduced by Barreto et al. [2014] called policy iteration based on stochastic factorization (PISF). Given $D^aK \approx P^a$ and $D^a\bar{r} \approx r^a$, PISF computes an approximation of the MDP’s value function, $\tilde{v} \approx v^*$, in $O(n)$ time per iteration; Barreto et al. [2014] have shown that $\|v^* - \tilde{v}\|_{\infty} \to 0$ as $\|P^a - D^aK\|_{\infty} \to 0$ and $\|r^a - D^a\bar{r}\|_{\infty} \to 0$ for all $a \in A$. EMSF computed the factorization used by PISF from the same data used by CNT. Since in blackjack the rewards are completely defined by the end state of a transition, we can compute $\bar{r} = K\bar{r}$, where $\bar{r}_i$ is an estimate of the reward associated with transitions ending in state $i$, $\bar{r}_i$. This is clear when we note that $r^a = P^{a\beta}K^{a\beta}$. Figure 3 compares the results obtained by CNT+PI and EMSF+PISF on blackjack. Both methods are contrasted with an agent that uses the same strategy used by the game’s dealer [Sutton and Barto, 1998]. Note how the results shown in Figure 3 reproduce the trend seen in Figure 2. In particular, after playing only 3000 hands, all the EMSF+PISF agents have already outperformed the dealer’s strategy, while CNT+PI needs twice as many hands to reach the same level of playing. The superior performance of EMSF+PISF is maintained up until 30000 hands played. These results show that the structural assumption underlying EMSF, a stochastic factorization, do arise in real applications—which in turn suggests that our algorithm can be useful in practice.
6 Related Work

Stochastic factorization is a particular case of nonnegative matrix factorization [Paatero and Tapper, 1994; Lee and Seung, 1999]. In fact, Cohen and Rothblum [1991] have shown that one can always derive a stochastic factorization from a nonnegative factorization of a stochastic matrix. In principle, thus, any algorithm designed for the latter can also be used to compute the former (see discussion in Barreto et al.’s [2014] paper). Note though that algorithms designed for nonnegative matrix factorization assume that we have access to the elements of P (or at least to a subset of them). In contrast, EMSF has no access to any element of P, only to data extracted from this matrix (this is possible precisely because of the extra structure assumed by stochastic factorization, since it is unclear how to extract data from a nonnegative matrix that is not stochastic).

As mentioned in the introduction, in the scenario considered here it is possible to start from an estimate of P and then use it to compute D and K through a nonnegative matrix factorization algorithm. Similar ideas have been explored by Finesso et al. [2010], Lakshminarayanan and Raich [2010], and Cybenko and Crespi [2011] in the context of HMM learning—except that instead of estimating P these authors estimate alternative matrix representations of the observation dynamics. A potential problem with applying these or similar approaches to learn a SFM is that the matrices representing the observation dynamics are at least $n \times n$, which amounts to estimating $O(n^2)$ parameters or more—exactly what EMSF is trying to avoid.

EMSF has clear similarities with the well known Baum-Welch algorithm, an EM method to compute the parameters of an HMM from data [Baum, 1972]. Given this similarity, and considering the connection between SFMs and HMMs, shown in Figure 1, it is natural to ask whether other methods to learn an HMM could also be adapted to compute a stochastic factorization (since the literature on this subject is vast, we redirect the reader to Cappé et al.’s [2005] book to serve as a starting point).

From a broader perspective, SFMs are a particular instance of latent variable models. Recently, there has been renewed interest in the development of algorithms to learn the parameters of such models, motivated by a reduction of the problem to a singular value decomposition of some matrix representing the observation dynamics [Hsu et al., 2009; Siddiqi et al., 2010; Song et al., 2010; Bailly, 2011; Balle et al., 2012]. Unlike EM, these so-called “spectral methods” always converge to a global optimum, and are also statistically consistent (under some reasonable assumptions). As with the algorithms that learn an HMM based on a nonnegative matrix factorization, the main difficulty in adapting spectral methods to learn a SFM is the fact that they build on the decomposition of a matrix that is at least $n \times n$, which means they require estimating at least $n^2$ parameters. In any case, the development of alternative methods to learn a SFM that overcome the limitations of EM is a promising topic for future research.

7 Conclusion

This paper introduced SFM, a probabilistic model that encompasses a stochastic factorization, and presented an expectation-maximization algorithm, EMSF, to compute the model’s parameters from data. Our algorithm has a single parameter, $m$, which can be seen as a practical mechanism to control the bias-variance trade off in the approximation of a transition matrix. This makes it possible to adjust the complexity of the model to the amount of data available; if the transition matrix is factorizable or nearly so, one can compute good approximations using less data than would be needed for direct maximum likelihood estimation. Our experiments suggest that factorizable transition matrices arise in practice.

One interesting direction for future research is to extend the ideas presented in this paper to continuous state spaces. Another promising line of investigation is the development of an incremental version of EMSF which makes it possible to incorporate new data into the model without the need to store the sample transitions already used [Khreich et al., 2012]. This extension would allow EMSF to scale to much larger datasets and also to be applied on-line, potentially using the model already constructed to guide the collection of new sample transitions [Kearns and Singh, 1998; Brafman and Tennenholtz, 2003]. Finally, another interesting direction for future research is the study of SFMs with factored dynamics, in which the model’s conditional probabilities can be represented by a dynamic Bayesian network [Guestrin et al., 2003]. This would bridge the gap between “factorizable” and “factored” models, potentially decreasing even further the number of samples needed to learn a transition matrix [Barreto et al., 2014].

Acknowledgments

The authors would like to thank Borja Pigem for helpful discussions regarding topics related to this work. Funding for this research was provided by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), grant 461739/2014-3, and by the NSERC Discovery grant program.
References


