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 characteristics 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 strategy, 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 decision 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, estimating 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 likelihood, 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 learning agent learn how to play the game of blackjack.

1 Introduction

Results from the supervised learning literature suggest 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 on a lower dimensional manifold [Farahmand, 2011]. Such regularities 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 fundamental 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 reinforcement learning [Barreto *et al.*, 2011].

So far, the stochastic factorization problem has been addressed in the literature mainly from an "algebraic" point of view: given a transition matrix, one must find two lower-order stochastic matrices whose multiplication approximates the original matrix as well as possible. In this paper we focus on learning a stochastic factorization when the transition matrix is not known, and we only have access to transitions sampled from it. One possible approach in this case is to use the samples to compute an estimate of the transition matrix, which takes use back to the scenario described above. However, this approach has the disadvantage that many entries may need to be estimated, perhaps 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 interpretation of a stochastic factorization and build on the resulting model to derive an expectation-maximization algorithm that achieves this goal.

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 errors [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, the proposed method is able to compute approximations of roughly the same quality using less data. To illustrate the practical utility of the proposed approach, 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 $A_{1:\tau}$ to refer to a sequence of variables $A_1, A_2, ..., A_\tau$. 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 $\mathbf{P} \in \mathbb{R}^{n\times q}$ 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 $\mathbf{P} \in \mathbb{R}^{n\times q}$, the relation $\mathbf{P} = \mathbf{DK}$ is called a* stochastic factorization *of $\mathbf{P}$ if $\mathbf{D} \in \mathbb{R}^{n\times m}$ and $\mathbf{K} \in \mathbb{R}^{m\times q}$ are also stochastic matrices. The integer $m > 0$ is the* order *of the factorization.*

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

Given a stochastic factorization of a transition matrix, $\mathbf{P} = \mathbf{DK}$, *swapping* the factors of the factorization yields another transition matrix $\bar{\mathbf{P}} = \mathbf{KD}$, potentially much smaller than the original, which retains the basic topology of $\mathbf{P}$—that is, the number of recurrent classes and their respective reducibilities and periodicities (see Barreto and Fragoso's [2011] article for formal definitions). Since $\bar{\mathbf{P}}$ can be considerably smaller than $\mathbf{P}$, 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

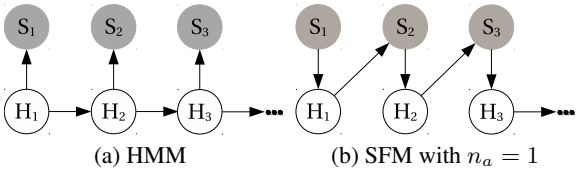
So far, an algebraic view of stochastic factorization has prevailed in the literature, with $\mathbf{P}$, $\mathbf{D}$, and $\mathbf{K}$ treated simply as matrices with a particular structure. In this paper we introduce a probabilistic interpretation of a stochastic factorization. In particular, we see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

3.1 Stochastic Factorization Model

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

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

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



(a) HMM (b) SFM with $n_a = 1$

Figure 1: Graphical models of HMMs and SFMs.

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

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

$$\begin{aligned}&\Pr(S_1, A_1, H_1, ..., S_{\tau-1}, A_{\tau-1}, H_{\tau-1}, S_\tau) = \\ &= \Pr(S_1)\textstyle\prod_{t=2}^{\tau}\Pr(S_t|H_{t-1}, A_{t-1}, ..., S_1)\times \\ &\quad\times\Pr(A_{t-1}|S_{t-1}, H_{t-2}, ..., S_1)\Pr(H_{t-1}|A_{t-1}, S_{t-1}, ..., S_1) \\ &= \Pr(S_1)\textstyle\prod_{t=2}^{\tau}\Pr(S_t|H_{t-1}, A_{t-1})\Pr(A_{t-1}|S_{t-1})\times \\ &\quad\times\Pr(H_{t-1}|S_{t-1}, A_{t-1}).\end{aligned} \tag{1}$$

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 $\mathrm{F} \equiv (\mathbf{D}^a, \mathbf{K}^a, \mathbf{\Pi}, \boldsymbol{\mu})$, where $\mathbf{D}^a \in \mathbb{R}^{n\times m}$ and $\mathbf{K}^a \in \mathbb{R}^{n\times m}$ are n_a stochastic matrices, one for each $a \in \mathcal{A}$, $\mathbf{\Pi} \in \mathbb{R}^{n\times n_a}$ is a stochastic matrix, and $\boldsymbol{\mu} \in \mathbb{R}^n$ is a distribution. Based on Assumptions (i), (ii), and (iii), the model represents the stochastic process $(S_1, A_1, H_1, S_2, A_2, H_2, ...)$ as $d^a_{ij} = \Pr(H_t = j|S_t = i, A_t = a)$, $k^a_{ij} = \Pr(S_{t+1} = j|H_t = i, A_t = a)$, $\pi_{ia} = \Pr(A_t = a|S_t = i)$, and $\mu_i = \Pr(S_1 = i)$.*

It may be instructive to revisit the stochastic-factorization trick in light of Definition 4. If $\mathbf{P}^a = \mathbf{D}^a\mathbf{K}^a$, it should be clear that $p^a_{ij} = \Pr(S_{t+1} = j|S_t = i, A_t = a)$. Similarly, if we apply the trick to obtain $\bar{\mathbf{P}}^a = \mathbf{K}^a\mathbf{D}^a$, then we have $\bar{p}^a_{ij} = \Pr(H_{t+1} = j|H_t = i, A_t = a)$.

Note that a SFM with $n_a = 1$ 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 S_t is sufficient to define the distribution of H_t.

3.2 Computing Probabilities

Recall that in a SFM only the variables S_t and A_t are observable. Let $z_{1:\tau} = (s_1, a_1, s_2, ..., a_{\tau-1}, s_\tau)$. In order to do inference with our model, we need to be able to compute the probability of a given sequence happening, that is, $\Pr(Z_{1:\tau} = z_{1:\tau}|\lambda)$, where $\lambda \equiv (\mathbf{D}^a, \mathbf{K}^a, \mathbf{\Pi}, \boldsymbol{\mu})$. 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 $\Pr(Z_{1:\tau} = z_{1:\tau}, H_t = i|\lambda)$, then we can compute the desired quantity by marginalizing over H_t. Given $z_{1:\tau}$, we define the "forward variable" $\alpha_i(t)$ as:

$$\alpha_i(t) = \Pr(Z_{1:t} = z_{1:t}, A_t = a_t, H_t = i|\lambda). \tag{2}$$

$$
\begin{aligned}
\alpha_i(1) &= \mu_{s_1}\pi_{s_1a_1}d^{a_1}_{s_1i} = \Pr(S_1=s_1)\Pr(A_1=a_1|S_1=s_1)\Pr(H_1=i|S_1=s_1,A_1=a_1)\\
&= \Pr(S_1=s_1,A_1=a_1)\Pr(H_1=i|S_1=s_1,A_1=a_1) = \Pr(S_1=s_1,A_1=a_1,H_1=i)
\end{aligned}
$$

$$
\begin{aligned}
\alpha_j(t+1) &= \textstyle\sum_i \alpha_i(t)k^{a_t}_{is_{t+1}}\pi_{s_{t+1}a_{t+1}}d^{a_{t+1}}_{s_{t+1}j} = \sum_i \Pr(Z_{1:t}=z_{1:t},A_t=a_t,H_t=i)\Pr(S_{t+1}=s_{t+1}|A_t=a_t,H_t=i)\times\\
&\quad\times\Pr(A_{t+1}=a_{t+1}|S_{t+1}=s_{t+1})\Pr(H_{t+1}=j|S_{t+1}=s_{t+1},A_{t+1}=a_{t+1})\\
&= \textstyle\sum_i \Pr(Z_{1:t}=z_{1:t},A_t=a_t,H_t=i,S_{t+1}=s_{t+1})\Pr(A_{t+1}=a_{t+1}|S_{t+1}=s_{t+1})\times\\
&\quad\times\Pr(H_{t+1}=j|S_{t+1}=s_{t+1},A_{t+1}=a_{t+1})\\
&= \textstyle\sum_i \Pr(Z_{1:t}=z_{1:t},A_t=a_t,H_t=i,S_{t+1}=s_{t+1},A_{t+1}=a_{t+1})\Pr(H_{t+1}=j|S_{t+1}=s_{t+1},A_{t+1}=a_{t+1})\\
&= \textstyle\sum_i \Pr(Z_{1:t}=z_{1:t},A_t=a_t,H_t=i,S_{t+1}=s_{t+1},A_{t+1}=a_{t+1},H_{t+1}=j)\\
&= \Pr(Z_{1:t+1}=z_{1:t+1},A_{t+1}=a_{t+1},H_{t+1}=j)
\end{aligned}
$$

$$
\beta_i(\tau-1) = k^{a_{\tau-1}}_{is_\tau} = \Pr(S_\tau=s_\tau|h_{\tau-1}=i,A_{\tau-1}=a_{\tau-1}) = \Pr(Z_{\tau:\tau}=z_{\tau:\tau}|H_{\tau-1}=i,A_{\tau-1}=a_{\tau-1})
$$

$$
\begin{aligned}
\beta_i(t-1) &= \textstyle\sum_j \beta_j(t)d^{a_t}_{s_tj}\pi_{s_ta_t}k^{a_{t-1}}_{i,s_t} = \sum_j \Pr(Z_{t+1:\tau}=z_{t+1:\tau}|H_t=j,A_t=a_t)\Pr(H_t=j|S_t=s_t,A_t=a_t)\times\\
&\quad\times\Pr(A_t=a_t|S_t=s_t)\Pr(S_t=s_t|H_{t-1}=i,A_{t-1}=a_{t-1})\\
&= \textstyle\sum_j \Pr(Z_{t+1:\tau}=z_{t+1:\tau}|H_t=j,A_t=a_t,S_t=s_t)\Pr(H_t=j|S_t=s_t,A_t=a_t)\times\\
&\quad\times\Pr(A_t=a_t|S_t=s_t)\Pr(S_t=s_t|H_{t-1}=i,A_{t-1}=a_{t-1})\\
&= \textstyle\sum_j \Pr(H_t=j,Z_{t+1:\tau}=z_{t+1:\tau}|S_t=s_t,A_t=a_t)\Pr(A_t=a_t|S_t=s_t)\Pr(S_t=s_t|H_{t-1}=i,A_{t-1}=a_{t-1})\\
&= \textstyle\sum_j \Pr(A_t=a_t,H_t=j,Z_{t+1:\tau}=z_{t+1:\tau}|S_t=s_t)\Pr(S_t=s_t|H_{t-1}=i,A_{t-1}=a_{t-1})\\
&= \textstyle\sum_j \Pr(A_t=a_t,H_t=j,Z_{t+1:\tau}=z_{t+1:\tau}|S_t=s_t,H_{t-1}=i,A_{t-1}=a_{t-1})\Pr(S_t=s_t|H_{t-1}=i,A_{t-1}=a_{t-1})\\
&= \textstyle\sum_j \Pr(S_t=s_t,A_t=a_t,H_t=j,Z_{t+1:\tau}=z_{t+1:\tau}|H_{t-1}=i,A_{t-1}=a_{t-1})\\
&= \Pr(S_t=s_t,A_t=a_t,Z_{t+1:\tau}=z_{t+1:\tau}|H_{t-1}=i,A_{t-1}=a_{t-1}) = \Pr(Z_{t:\tau}=z_{t:\tau}|H_{t-1}=i,A_{t-1}=a_{t-1})
\end{aligned}
$$

$$
\begin{aligned}
\alpha_i(t)\beta_i(t) &= \Pr(Z_{1:t}=z_{1:t},A_t=a_t,H_t=i)\Pr(Z_{t+1:\tau}=z_{t+1:\tau}|H_t=i,A_t=a_t)\\
&= \Pr(Z_{1:t}=z_{1:t},A_t=a_t,H_t=i)\Pr(Z_{t+1:\tau}=z_{t+1:\tau}|H_t=i,A_t=a_t,Z_{1:t}=z_{1:t}) = \Pr(Z_{1:\tau}=z_{1:\tau},H_t=i).
\end{aligned}
$$

Table 1: Computing probabilities in a SFM; the dependence on λ is omitted to improve readability.

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

$$
\begin{aligned}
\alpha_i(1) &= \mu_{s_1}\pi_{s_1a_1}d^{a_1}_{s_1i},\\
\alpha_j(t+1) &= \textstyle\sum_i \alpha_i(t)k^{a_t}_{is_{t+1}}\pi_{s_{t+1}a_{t+1}}d^{a_{t+1}}_{s_{t+1}j}.
\end{aligned}
$$

We now define the "backward variable" $\beta_i(t)$ as:

$$
\beta_i(t) = \Pr(Z_{t+1:\tau}=z_{t+1:\tau}|H_t=i,A_t=a_t,\lambda). \tag{3}
$$

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

$$
\begin{aligned}
\beta_i(\tau-1) &= k^{a_{\tau-1}}_{is_\tau},\\
\beta_i(t-1) &= \textstyle\sum_j \beta_j(t)d^{a_t}_{s_tj}\pi_{s_ta_t}k^{a_{t-1}}_{i,s_t}.
\end{aligned}
$$

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,

$$
\alpha_i(t)\beta_i(t) = \Pr(Z_{1:\tau}=z_{1:\tau},H_t=i|\lambda). \tag{4}
$$

The equality above is also justified in Table 1.

Scaling

Using the variables $\alpha_i(t)$ and $\beta_i(t)$ we can compute $\Pr(Z_{1:\tau},h_t=i|\lambda)$, 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 now 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 $w_t=\sum_i\alpha_i(t)$. Starting with $\hat{\alpha}_i(1)=\alpha_i(1)/w_1$, we will apply the following recursion, for $t\geq 2$:

$$
\begin{aligned}
\alpha'_j(t) &= \textstyle\sum_i \hat{\alpha}_i(t-1)k^{a_{t-1}}_{is_t}\pi_{s_ta_t}d^{a_t}_{s_tj},\\
\hat{\alpha}_j(t) &= \alpha'_j(t)/w'_t,
\end{aligned} \tag{5}
$$

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

$$
\hat{\alpha}_i(t) = \prod_{j=1}^{t}(w'_j)^{-1}\alpha_i(t) = w_t^{-1}\alpha_i(t), \tag{6}
$$

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

$$
\begin{aligned}
\Pr(Z_{1:\tau}|\lambda) &= \sum_{i=1}^{m}\alpha_i(\tau-1)\beta_i(\tau-1)\\
&= \prod_{j=1}^{\tau-1}w'_j\sum_{i=1}^{m}\hat{\alpha}_i(\tau-1)k^{a_{\tau-1}}_{is_\tau},
\end{aligned} \tag{7}
$$

which means that, as long as the variables w'_j 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 $\Pr(Z_{1:\tau}|\lambda)$, it will be convenient to also define their scaled version. Starting with $\hat{\beta}_i(\tau-1)=\beta_i(\tau-1)$ and $\hat{\beta}_i(\tau-2)=\beta_i(\tau-2)/w_{\tau-1}$, we will apply the following recursion, for $t<\tau-2$:

$$
\begin{aligned}
\beta'_i(t-1) &= \textstyle\sum_j \hat{\beta}_j(t)d^{a_t}_{s_tj}\pi_{s_ta_t}k^{a_{t-1}}_{i,s_t},\\
\hat{\beta}_i(t-1) &= \beta'_i(t-1)/w'_t.
\end{aligned} \tag{8}
$$

Note that, since the magnitude of $\beta'_i(t-1)$ and $\alpha'_i(t)$ is comparable, the computation of $\hat{\beta}_i(t)$ will also be numerically stable. One can show that

$$\hat{\beta}_i(t) = \prod_{j=t+1}^{\tau-1} (w'_j)^{-1} \beta_i(t). \tag{9}$$

From (6) and (9) we know that $\hat{\alpha}_i(t)\hat{\beta}_i(t) = \prod_{j=1}^{\tau-1}(w'_j)^{-1}\alpha_i(t)\beta_i(t)$, which, combined with (4), implies that

$$\frac{\hat{\alpha}_i(t)\hat{\beta}_i(t)}{\sum_i \hat{\alpha}_i(t)\hat{\beta}_i(t)} = \frac{\alpha_i(t)\beta_i(t)}{\sum_i \alpha_i(t)\beta_i(t)} = \Pr(H_t = i | Z_{1:\tau}, \lambda). \tag{10}$$

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_a transition matrices $\mathbf{P}^a$, instead of estimating the matrices $\mathbf{P}^a$ directly, we estimate a SFM $\mathrm{F} \equiv (\mathbf{D}^a, \mathbf{K}^a, \mathbf{\Pi}, \boldsymbol{\mu})$. By doing so, we automatically have approximations $\mathbf{D}^a\mathbf{K}^a \approx \mathbf{P}^a$.

The strategy above has two benefits. First, by estimating $\mathbf{D}^a$ and $\mathbf{K}^a$ instead of $\mathbf{P}^a$, 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 multiplication $\mathbf{D}^a\mathbf{K}^a$: instead, we compute $\bar{\mathbf{P}}^a = \mathbf{K}^a\mathbf{D}^a$ and use this smaller matrix in place of $\mathbf{P}^a$.

In this section we discuss how to compute a SFM directly from data. Since it is a latent-variable model, we resort to the well know 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

$$\mathcal{Q}(\lambda, \lambda') = \sum_{h_{1:\tau} \in \mathcal{H}_{1:\tau}} \log \Pr(h_{1:\tau}, Z_{1:\tau}|\lambda) \Pr(h_{1:\tau}|Z_{1:\tau}, \lambda'). \tag{11}$$

In the "E" step we use the current parameter values λ' to compute the expectation above; in the "M" step we maximize the expectation we computed with respect to λ. 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

$$\Pr(h, Z|\lambda) = \mu_{s_1} \prod_{t=1}^{\tau-1} \pi_{s_t a_t} d^{a_t}_{s_t h_t} k^{a_t}_{h_t s_{t+1}}. \tag{12}$$

Substituting (12) in (11), the $\mathcal{Q}$ function becomes

$$\begin{aligned}\mathcal{Q}(\lambda, \lambda') = &\sum_{h\in\mathcal{H}} \log \mu_{s_1} \Pr(h|Z, \lambda') + \\ &+ \sum_{h\in\mathcal{H}} \left(\sum_{t=1}^{\tau-1} \log \pi_{s_t a_t}\right) \Pr(h|Z, \lambda') + \\ &+ \underbrace{\sum_{h\in\mathcal{H}} \left(\sum_{t=1}^{\tau-1} \log d^{a_t}_{s_t h_t}\right) \Pr(h|Z, \lambda')}_{\Delta_D} + \\ &+ \underbrace{\sum_{h\in\mathcal{H}} \left(\sum_{t=1}^{\tau-1} \log k^{a_t}_{h_t s_{t+1}}\right) \Pr(h|Z, \lambda')}_{\Delta_K}\end{aligned}$$

(recall that here $\mathcal{H}$ represents the space of all possible sequences h of length τ). We will focus on the third term of the equation above, tagged as "Δ_D," to show how to derive update equations for the elements of the matrices $\mathbf{D}^a$; 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 $\log d^u_{ij}$ appearing in Δ_D will multiply $\Pr(h|Z, \lambda')$ if and only if the t^{th} element in h is j—that is, d^u_{ij} will multiply $\sum_{h\in\mathcal{H}} \Pr(h|Z, \lambda')\mathbf{1}\{h_t = j\} = \Pr(H_t = j|Z, \lambda')$ (here $\mathbf{1}\{\cdot\}$ is the indicator function). Thus, we can rewrite Δ_D as:

$$\Delta_D = \sum_{u=1}^{n_a}\sum_{i=1}^{n}\sum_{j=1}^{m} \log d^u_{ij} \sum_{t=1}^{\tau-1} \Pr(H_t = j|Z, \lambda')\mathbf{1}\{s_t = i, a_t = u\}.$$

We want to maximize $\mathcal{Q}(\lambda, \lambda')$ with respect to λ. One way to do so is to add the Lagrange multipliers κ^u_i to account for the constraints $\sum_j d^u_{ij} = 1$, which leads to

$$\Delta'_D = \Delta_D - \sum_{u=1}^{n_a}\sum_{i=1}^{n} \kappa^u_i \left(\sum_{j=1}^{m} d^u_{ij} - 1\right).$$

Now, if we make $\partial\Delta'_D/\partial d^u_{ij} = 0$, we have a necessary condition for d^u_{ij} to be an extreme point of $\mathcal{Q}$. Since

$$\frac{\partial\Delta'_D}{\partial d^u_{ij}} = \frac{1}{d^u_{ij}} \sum_{t=1}^{\tau-1} \Pr(H_t = j|Z, \lambda')\mathbf{1}\{s_t = i, a_t = u\} - \kappa^u_i,$$

we know that

$$d^u_{ij} = \left(\sum_{t=1}^{\tau-1} \Pr(H_t = j|Z, \lambda')\mathbf{1}\{s_t = i, a_t = u\}\right) / \kappa^u_i. \tag{13}$$

If we sum (13) over j, using the fact that $\sum_j d^u_{ij} = 1$, we obtain $\kappa^u_i = \sum_{t=1}^{\tau-1} \mathbf{1}\{s_t = i, a_t = u\}$. Substituting κ^u_i back in (13), we have the following update rule for d^u_{ij}:

$$d^u_{ij} = \frac{\sum_{t=1}^{\tau-1} \Pr(H_t = j|Z, \lambda')\mathbf{1}\{s_t = i, a_t = u\}}{\sum_{t=1}^{\tau-1} \mathbf{1}\{s_t = i, a_t = u\}}. \tag{14}$$

If we follow the same steps as above replacing Δ_D with Δ_K we arrive at the following update equation for k^u_{ij}:

$$k^u_{ij} = \frac{\sum_{t=1}^{\tau-1} \Pr(H_t = i|Z, \lambda')\mathbf{1}\{s_{t+1} = j, a_t = u\}}{\sum_{t=1}^{\tau-1} \Pr(H_t = i|Z, \lambda')\mathbf{1}\{a_t = u\}}. \tag{15}$$

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, $z^l_{1:\tau_l} = \{s^l_1, a^l_1, ..., a^l_{\tau_l-1}, s^l_{\tau_l}\}$, with $l = 1, 2, ..., c$. In this case we have the following update rules:

$$d^u_{ij} = \frac{\sum_{l=1}^{c}\sum_{t=1}^{\tau_l-1}\Pr(H_t = j|Z = z^l, \lambda')\mathbf{1}\{s^l_t = i, a^l_t = u\}}{\sum_{l=1}^{c}\sum_{t=1}^{\tau_l-1}\mathbf{1}\{s^l_t = i, a^l_t = u\}}. \tag{16}$$

$$k^u_{ij} = \frac{\sum_{l=1}^{c}\sum_{t=1}^{\tau_l-1}\Pr(H_t = i|Z = z^l, \lambda')\mathbf{1}\{s^l_{t+1} = j, a^l_t = u\}}{\sum_{l=1}^{c}\sum_{t=1}^{\tau_l-1}\Pr(H_t = i|Z = z^l, \lambda')\mathbf{1}\{a^l_t = u\}}. \tag{17}$$

In principle, we could follow the steps above to derive update equations for μ_i and π_{iu}. 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:

$$\mu_i = \frac{\sum_{l=1}^{c}\mathbf{1}\{s^l_1 = i\}}{c} \text{ and } \pi_{iu} = \frac{\sum_{l=1}^{c}\sum_{t=1}^{\tau_l-1}\mathbf{1}\{s^l_t = i, a^l_t = u\}}{\sum_{l=1}^{c}\sum_{t=1}^{\tau_l-1}\mathbf{1}\{s^l_t = i\}}. \tag{18}$$

Since the quantities appearing in (18) do not depend on the parameters of the SFM, the estimates of $\boldsymbol{\mu}$ and $\boldsymbol{\Pi}$ will be fixed throughout the EM iterations. Note that in scenarios where we have control over the collection of data we generally know $\boldsymbol{\mu}$ and $\boldsymbol{\Pi}$, 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 $\mathbf{D}$ or a single $\mathbf{K}$ (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 $\mathbf{K}$, for example, corresponds to replacing Assumption (ii) with $\Pr(H_t|A_t, S_t, H_{t-1}, ..., S_1) = \Pr(H_t|S_t)$. The resulting update equation will be expression (17) without the restriction $a^l_t = u$.

4.2 Algorithm

Algorithm 1 shows a step by step description of the proposed method, which we call EMSF. The pseudo-code uses two matrices to represent the forward and backward variables: $\boldsymbol{\alpha} \in \mathbb{R}^{\tau-1\times m}$, where $\alpha_{ti} = \hat{\alpha}_i(t)$, and $\boldsymbol{\beta} \in \mathbb{R}^{\tau-1\times m}$, where $\beta_{ti} = \hat{\beta}_i(t)$. For each sequence $z^l_{1:\tau_l}$, EMSF first computes $\boldsymbol{\alpha}$ using (5) and then computes $\boldsymbol{\beta}$ using (8). Then, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are multiplied element-wise, giving rise to matrix $\mathbf{C} \in \mathbb{R}^{\tau-1\times m}$. After the rows of $\mathbf{C}$ have been normalized, as shown in line 39 of Algorithm 1, we have $c_{ti} = \Pr(H_t = i|Z_{1:\tau_l} = z^l, \lambda')$ (see (10)). These values are then used to compute the numerator of update equations (16) and (17), which are accumulated in the auxiliary matrices $\hat{\mathbf{D}}^a \in \mathbb{R}^{n\times m}$ and $\hat{\mathbf{K}}^a \in \mathbb{R}^{m\times n}$. After all the sequences have been processed, the rows of $\hat{\mathbf{D}}^a$ and $\hat{\mathbf{K}}^a$ 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 $\mathbf{D}^a$

Algorithm 1 EMSF

$\{z^1_{1:\tau_1}, z^2_{1:\tau_2}, ..., z^c_{1:\tau_c}\}$ ▷ Data
$m \in \mathbb{N}$ ▷ Order of the factorization
1: **Input:** $\boldsymbol{\mu} \in \mathbb{R}^n$ ▷ Initial distribution
$\boldsymbol{\Pi} \in \mathbb{R}^{n\times|A|}$ ▷ Exploration policy
$\epsilon > 0$ ▷ Stop criterion
2: **Output:** $\mathbf{D}^a\mathbf{K}^a \approx \mathbf{P}^a$, for $a \in \mathcal{A}$
3:
4: **for** each $a \in \mathcal{A}$ **do**
5: $\mathbf{D}^a \leftarrow$ random stochastic matrix in $\mathbb{R}^{n\times m}$
6: $\mathbf{K}^a \leftarrow$ random stochastic matrix in $\mathbb{R}^{m\times n}$
7:
8: $\mathfrak{L} \leftarrow 0$; $\mathfrak{L}' \leftarrow \infty$ ▷ $\mathfrak{L}$ is the data log likelihood
9: **while** $|\mathfrak{L} - \mathfrak{L}'| > \epsilon$ **do**
10: $\mathfrak{L}' \leftarrow \mathfrak{L}$; $\mathfrak{L} \leftarrow 0$
11:
12: **for** each $a \in \mathcal{A}$ **do** $\hat{\mathbf{D}}^a \leftarrow \mathbf{0} \in \mathbb{R}^{n\times m}$; $\hat{\mathbf{K}}^a \leftarrow \mathbf{0} \in \mathbb{R}^{m\times n}$
13:
14: **for** $l \leftarrow 1$ **to** c **do**
15: $\tau \leftarrow \tau_l$
16: $s \leftarrow \{s^l_1, s^l_2, ..., s^l_\tau\}$ ▷ Extract $s^l_{i:\tau_l}$ from $z^l_{1:\tau_l}$
17: $a \leftarrow \{a^l_1, a^l_2, ..., a^l_{\tau-1}\}$ ▷ Extract $a^l_{i:\tau_l-1}$ from $z^l_{1:\tau_l}$
18: $\boldsymbol{\alpha} \leftarrow \mathbf{0} \in \mathbb{R}^{\tau-1\times m}$
19: $\boldsymbol{\beta} \leftarrow \mathbf{0} \in \mathbb{R}^{\tau-1\times m}$
20: $\mathbf{w}' \leftarrow \mathbf{0} \in \mathbb{R}^{\tau-1}$
21: ▷ Compute $\boldsymbol{\alpha}$
22:
23: $(\boldsymbol{\alpha})_1 \leftarrow \mu_1\pi_{s_1a_1}(\mathbf{D}^{a_1})_{s_1}$ ▷ $(\boldsymbol{\alpha})_i$ is $\boldsymbol{\alpha}$'s i^{th} row
24: $w'_1 \leftarrow \sum_i \alpha_{1i}$
25: $(\boldsymbol{\alpha})_1 \leftarrow (\boldsymbol{\alpha})_1/w'_1$
26: **for** $t \leftarrow 2$ **to** $\tau - 1$ **do**
27: $(\boldsymbol{\alpha})_t \leftarrow \left(\sum_i \alpha_{(t-1)i}k^{a_{t-1}}_{is_t}\right)\pi_{s_ta_t}(\mathbf{D}^{a_t})_{s_t}$
28: $w'_t \leftarrow \sum_i \alpha_{ti}$
29: $(\boldsymbol{\alpha})_t \leftarrow (\boldsymbol{\alpha})_t/w'_t$
30: ▷ Compute $\boldsymbol{\beta}$
31:
32: $(\boldsymbol{\beta})_{\tau-1} \leftarrow (\mathbf{K}^{a_{\tau-1}})^{s_\tau}$ ▷ $(\mathbf{K})^i$ is $\mathbf{K}$'s i^{th} column
33: **for** $t \leftarrow \tau - 2$ **to** 1 **do**
34: $(\boldsymbol{\beta})_t \leftarrow \sum_i \beta_{(t+1)i}d^{a_{t+1}}_{(s_{t+1})i}\pi_{(s_{t+1})(a_{t+1})}(\mathbf{K}^{a_t})^{s_{t+1}}$
35: $(\boldsymbol{\beta})_t \leftarrow (\boldsymbol{\beta})_t/w'_{t+1}$
36: ▷ Update $\hat{\mathbf{D}}^a$ and $\hat{\mathbf{K}}^a$
37:
38: $\mathbf{C} \leftarrow \boldsymbol{\alpha} \otimes \boldsymbol{\beta}$ ▷ Element-wise multiplication
39: **for** $i \in 1, 2, ..., \tau - 1$ **do** $(\mathbf{C})_i \leftarrow (\mathbf{C})_i/\sum_j c_{ij}$
40: **for** $t \leftarrow 1$ **to** $\tau - 1$ **do**
41: $(\hat{\mathbf{D}}^{a_t})_{s_t} \leftarrow (\hat{\mathbf{D}}^{a_t})_{s_t} + (\mathbf{C})_t$
42: $(\hat{\mathbf{K}}^{a_t})^{s_{t+1}} \leftarrow (\hat{\mathbf{K}}^{a_t})^{s_{t+1}} + (\mathbf{C})_t$
43:
44: ▷ Update $\mathfrak{L}$ based on (7)
45:
46: $\mathfrak{L} \leftarrow \mathfrak{L} + \sum_i \log w'_i + \log\left((\boldsymbol{\alpha})_{\tau-1}(\mathbf{K}^{a_{\tau-1}})^{s_\tau}\right)$
47:
48: ▷ Update $\mathbf{D}^a$ and $\mathbf{K}^a$
49:
50: **for** each $a \in \mathcal{A}$ **do**
51: **for** $i \in 1, 2, ..., n$ $(\hat{\mathbf{D}}^a)_i \leftarrow (\hat{\mathbf{D}}^a)_i/\sum_j \hat{d}^a_{ij}$
52: **for** $i \in 1, 2, ..., m$ $(\hat{\mathbf{K}}^a)_i \leftarrow (\hat{\mathbf{K}}^a)_i/\sum_j \hat{k}^a_{ij}$
53: $\mathbf{D}^a \leftarrow \hat{\mathbf{D}}^a$; $\mathbf{K}^a \leftarrow \hat{\mathbf{K}}^a$
54:
55: **return** $\mathbf{D}^a$ and $\mathbf{K}^a$, with $a \in \mathcal{A}$

and $\mathbf{K}^a$. 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 $\mathcal{L}$. The solution the algorithm converges to is determined by the initial values of $\mathbf{D}^a$ and $\mathbf{K}^a$. 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 $\mathbf{D}^a\mathbf{K}^a \approx \mathbf{P}^a$ (see discussion in Section 5) [Hastie *et al.*, 2002; Bishop, 2006].

As for computational complexity, each iteration of EMSF is $O(\tau m)$, where $\tau = \sum_l \tau_l$. Thus, the cost of the algorithm is not a direct function of n, the size of the matrices $\mathbf{P}^a$ being estimated. Of course, n will play a role in the definition of an appropriate value for τ, since EMSF estimates $2n_a nm$ 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 $\mathbf{P} \in \mathbb{R}^{100\times100}$, with $\mathrm{srk}(\mathbf{P}) = 20$, and tried to recover them with EMSF using different values for m. We then compared EMSF's results with those obtained by directly estimating $\mathbf{P}$ 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 $\mathcal{L}$, EMSF's approximation error stagnates at a positive value, even when $m = \mathrm{srk}(\mathbf{P})$, 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 τ. EMSF reduces the estimation error by enforcing a specific structure on $\mathbf{P}$—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 $\mathrm{srk}(\mathbf{P})$, 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.

It is clear then that EMSF can be seen as a strategy to regularize the approximation of $\mathbf{P}$ in order to reduce the number of transitions needed for learning. An interesting question is whether the structure imposed on $\mathbf{P}$ by our algorithm really arises in practice. In order to answer this question, we used EMSF to help a reinforcement learning agent learn how to play 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 in the following way. First, using CNT we computed estimates of both the transition matrices $\mathbf{P}^a$ and the expected-reward vectors $\mathbf{r}^a$ of the Markov decision process (MDP) describing the game. Policy iteration (PI) was then used with the resulting model to compute a policy, which was evaluated on 10^6 games of blackjack.

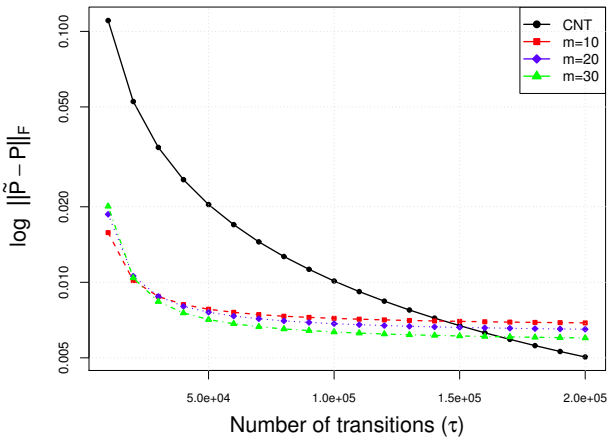


Figure 2: $\| \mathbf{P} - \tilde{\mathbf{P}} \|_{\mathrm{F}}$, where $\| \cdot \|_{\mathrm{F}}$ is the Frobenius norm and $\tilde{\mathbf{P}}$ is an estimate of $\mathbf{P}$. Here $\mathbf{P} = \mathbf{DK}$, with $\mathbf{D} \in \mathbb{R}^{100\times20}$ and $\mathbf{K} \in \mathbb{R}^{20\times100}$ generated by sampling their elements from a uniform distribution and then normalizing. Results were averaged over 50 runs.

In order to evaluate EMSF we used an algorithm introduced by Barreto *et al.* [2014] called policy iteration based on stochastic factorization (PISF). Given $\mathbf{D}^a\mathbf{K} \approx \mathbf{P}^a$ and $\mathbf{D}^a\bar{\mathbf{r}} \approx \mathbf{r}^a$, PISF computes an approximation of the MDP's value function, $\tilde{\mathbf{v}} \approx \mathbf{v}^*$, in $O(n)$ time per iteration; Barreto *et al.* [2014] have shown that $\| \mathbf{v}^* - \tilde{\mathbf{v}} \|_\infty \to 0$ as $\|\mathbf{D}^a\mathbf{K} - \mathbf{P}^a\|_\infty \to 0$ and $\|\mathbf{D}^a\bar{\mathbf{r}} - \mathbf{r}^a\|_\infty \to 0$ for all $a \in \mathcal{A}$. We used EMSF to compute the factorization used by PISF. Since in blackjack the rewards are completely defined by the end state of a transition, we can compute $\bar{\mathbf{r}}$ as $\bar{\mathbf{r}} = \mathbf{K}\hat{\mathbf{r}}$, where $\hat{r}_i$ is an estimate of the reward associated with transitions ending in state i. This is clear when we note that $\mathbf{r}^a = \mathbf{P}^a\hat{\mathbf{r}} \approx \mathbf{D}^a\mathbf{K}\hat{\mathbf{r}}$.

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 games, all the EMSF+PISF agents have already outperformed the dealer's strategy, while CNT+PI needs twice as many games to reach the same level of playing. The superior performance of EMSF+PISF is maintained up until 300000 games 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

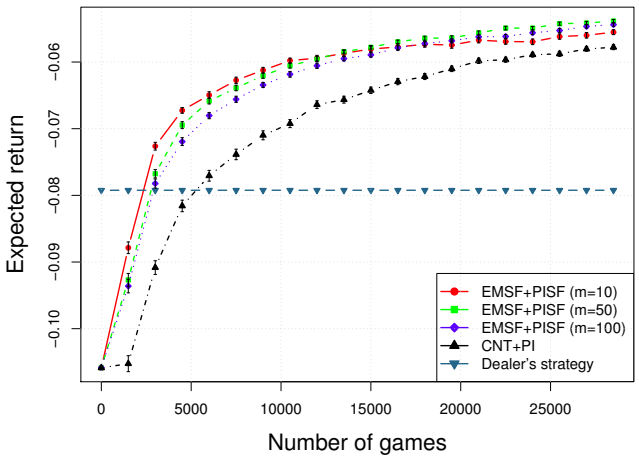


Figure 3: Results on the game of blackjack. The values correspond to the expected return obtained by each agent, estimated on 10^6 games. Error bars represent one standard error over 100 runs.

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 $\mathbf{P}$ (or at least to a subset of them). In contrast, EMSF has no access to any element of $\mathbf{P}$, only to data extracted from this matrix (this is possible precisely because of the extra structure assumed by stochastic factorization, since it is not clear 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 $\mathbf{P}$ and then use it to compute $\mathbf{D}$ and $\mathbf{K}$ 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 $\mathbf{P}$ 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).

Looking at SFMs from a broader perspective, they 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 to develop an incremental version of EMSF which makes it possible to incorporate new data to 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 investigations is the study of SFMs with factored dynamics, in which the model's 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 B. Pigem for helpful discussions regarding 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

[Bailly, 2011] Raphael Bailly. Quadratic weighted automata: Spectral algorithm and likelihood maximization. *Journal of Machine Learning Research*, 20:147–162, 2011.

[Balle *et al.*, 2012] Borja Balle, Ariadna Quattoni, and Xavier Carreras. Local loss optimization in operator models: A new insight into spectral learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1879–1886, 2012.

[Barreto and Fragoso, 2011] André M. S. Barreto and Marcelo D. Fragoso. Computing the stationary distribution of a finite Markov chain through stochastic factorization. *SIAM Journal on Matrix Analysis and Applications*, 32:1513–1523, 2011.

[Barreto *et al.*, 2011] André M. S. Barreto, Doina Precup, and Joelle Pineau. Reinforcement learning using kernel-based stochastic factorization. In *Advances in Neural Information Processing Systems (NIPS)*, pages 720–728, Granada, Spain, 2011.

[Barreto *et al.*, 2014] André M. S. Barreto, Joelle Pineau, and Doina Precup. Policy iteration based on stochastic factorization. *Journal of Artificial Intelligence Research*, 50:763–803, 2014.

[Baum, 1972] Leonard E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes. *Inequalities*, 3:1–8, 1972.

[Bilmes, 1998] Jeff A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical report, International Computer Science Institute, Berkley, CA, 1998.

[Bishop, 2006] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag Inc., 2006.

[Brafman and Tennenholtz, 2003] Ronen I. Brafman and Moshe Tennenholtz. R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2003.

[Cappé *et al.*, 2005] Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models*. Springer Series in Statistics. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[Cohen and Rothblum, 1991] Joel E. Cohen and Uriel G. Rothblum. Nonnegative ranks, decompositions and factorizations of nonnegative matrices. *Linear Algebra and its Applications*, 190:149–168, 1991.

[Cybenko and Crespi, 2011] G. Cybenko and V. Crespi. Learning hidden Markov models using nonnegative matrix factorization. *IEEE Transactions on Information Theory*, 57(6):3963–3970, 2011.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society: Series B*, 39:1–38, 1977.

[Farahmand, 2011] Amir-massoud Farahmand. *Regularization in reinforcement learning*. PhD thesis, University of Alberta, 2011.

[Finesso *et al.*, 2010] Lorenzo Finesso, Angela Grassi, and Peter Spreij. Approximation of stationary processes by hidden markov models. *Mathematics of Control, Signals, and Systems*, 22(1):1–22, 2010.

[Guestrin *et al.*, 2003] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[Györfi *et al.*, 2002] László Györfi, Michael Kohler, Adam Krzyżak, and Harro Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer Verlag, New York, 2002.

[Hastie *et al.*, 2002] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2002.

[Hsu *et al.*, 2009] D. Hsu, S. M. Kakade, and T. Zhang. A spectral algorithm for learning hidden Markov models. In *Proceedings of the Conference on Learning Theory (COLT)*, 2009.

[Kearns and Singh, 1998] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1998.

[Khreich *et al.*, 2012] Wael Khreich, Eric Granger, Ali Miri, and Robert Sabourin. A survey of techniques for incremental learning of HMM parameters. *Information Sciences*, 197:105–130, 2012.

[Lakshminarayanan and Raich, 2010] B. Lakshminarayanan and R. Raich. Non-negative matrix factorization for parameter estimation in hidden Markov models. In *Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, 2010.

[Lee and Seung, 1999] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[Paatero and Tapper, 1994] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5:111–126, 1994.

[Rabiner, 1989] Lawrence R Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. errata at `http://alumni.media.mit.edu/~rahimi/rabiner/rabiner-errata/rabiner-errata.html`.

[Siddiqi *et al.*, 2010] S. M. Siddiqi, B. Boots, and G. J. Gordon. Reduced-rank hidden markov models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[Song *et al.*, 2010] Le Song, Byron Boots, Sajid M Siddiqi, Geoffrey J Gordon, and Alex J Smola. Hilbert space embeddings of hidden markov models. In *Proceedings of the 27th international conference on machine learning (ICML)*, pages 991–998, 2010.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.