Journal of Global Optimization
https://doi.org/10.1007/s10898-022-01198-0

Fitting feature-dependent Markov chains

Shane Barratt[1] · **Stephen Boyd**[1]

Received: 13 September 2021 / Accepted: 30 May 2022

Abstract
We describe a method for fitting a Markov chain, with a state transition matrix that depends on a feature vector, to data that can include missing values. Our model consists of separate logistic regressions for each row of the transition matrix. We fit the parameters in the model by maximizing the log-likelihood of the data minus a regularizer. When there are missing values, the log-likelihood becomes intractable, and we resort to the expectation-maximization (EM) heuristic. We illustrate the method on several examples, and describe our efficient Python open-source implementation.

Keywords Markov chains · Convex optimization · Expectation maximization · Missing data

1 Introduction

Time-varying Markov chain. We consider data consisting of a discrete state $s_t \in \{1, \ldots, n\}$ and a feature vector $x_t \in \mathbf{R}^m$, where $t = 1, 2, \ldots$ denotes time (also called the period or epoch). We model the discrete state sequence as a time-varying Markov chain, *i.e.*,

$$\mathbf{Prob}(s_{t+1} = j \mid s_t = i) = (P_t)_{ij}, \quad i, j = 1, \ldots, n,$$

where $P_t \in \mathcal{P}$ is the transition matrix at time t, $\mathcal{P} = \{P \in \mathbf{R}_+^{n \times n} \mid P\mathbf{1} = \mathbf{1}\}$ is the set of (row) stochastic matrices, and $\mathbf{1}$ is the vector with all entries one. The ith row of P_t gives the probability distribution of the successor state s_{t+1} given $s_t = i$. The *Markov property* is the fact that under this model, given s_t, the future states s_τ, $\tau > t$, are independent of the past states s_η, $\eta < t$.
Transition matrix predictor. We model the transition matrix P_t as a function of the current feature vector x_t, *i.e.*,

✉ Stephen Boyd
boyd@stanford.edu

[1] Stanford University, Stanford, United States

Published online: 17 June 2022

$$P_t = \phi(x_t),$$

where $\phi : \mathbf{R}^m \to \mathcal{P}$ is the transition matrix predictor, so our overall model for s_t is a feature-dependent Markov chain. We let $\phi_i : \mathbf{R}^m \to \mathbf{R}^n$ denote the ith row of ϕ (transposed), *i.e.*,

$$\phi(x) = \begin{bmatrix} \phi_1(x)^T \\ \vdots \\ \phi_n(x)^T \end{bmatrix}.$$

We interpret $\phi_i(x_t)$ as the feature-dependent probability distribution of the successor state s_{t+1}, when $s_t = i$. The predictor ϕ is parametrized by a set of parameters, which we denote by $\theta \in \Theta$. We will often suppress this dependence in our notation.

Logistic predictor. We will focus on the logistic predictor, which has the form

$$\phi_i(x) = \frac{\exp y_i}{\mathbf{1}^T \exp y_i}, \qquad y_i = A_i x + b_i, \quad i = 1, \ldots, n, \tag{1}$$

where exp applies elementwise to the vector $y_i \in \mathbf{R}^n$, and $\mathbf{1}$ is the vector with all entries one. The parameters in this predictor are $\theta = (A_1, \ldots, A_n, b_1, \ldots, b_n)$, where $A_i \in \mathbf{R}^{n \times m}$, $b_i \in \mathbf{R}^n$, $i = 1, \ldots, n$. Thus we have a separate logistic regression model for each row of the transition matrix, *i.e.*, each value of the current state [12]. The total number of (scalar) parameters in our model is $n^2(m + 1)$. The parameters are redundant, since adding any multiple of $\mathbf{1}$ to b_i yields the same probability distribution ϕ_i. We can handle this redundancy by requiring that $\mathbf{1}^T b_i = 0$, or just letting regularization, described below, take care of it.

Prior sparsity pattern. The methods we describe are readily extended to handle the case when some state transitions are known to be impossible, which is the same as saying that P_t has some given sparsity pattern. This is handled by modifying $\phi_i(x)$ to be zero in the entries to which state i cannot transition, and using a logistic predictor for only the allowed next states. This means that the height of A_i and b_i is no longer n, but $n_i \leq n$, the number of allowed successor states of state i. The notation for this extension gets complicated, however, so we describe the simpler case when P_t is full.

Using the state as a feature. In our predictor (1) we have a separate logistic model ϕ_i for each current state i. We mention here a simpler predictor that instead uses the current state as a feature in a single logistic model. The simpler predictor has the form

$$\phi_i(x) = \frac{\exp y}{\mathbf{1}^T \exp y}, \qquad y = Ax + b_i,$$

with parameters $A \in \mathbf{R}^{n \times m}$ and $b_1, \ldots, b_n \in \mathbf{R}^n$. The total number of scalar parameters in this model is $n(m + n)$, compared to $n^2(m + 1)$ for our predictor (1).

In this predictor the current state enters $\phi_i(x)$, the next state distribution conditioned on $s_t = i$, in a very specific and constrained way. At the same time the feature vector x enters each $\phi_i(x)$ in a similar way. For example, each ϕ_i has the same monotonicity with respect to each feature x_j. While we can imagine cases in which this simpler predictor would perform well, we have observed that our predictor, with an independent logistic regression for each current state, performs better.

Training data. We consider two types of training data: full data, and missing data. In the case of full data, our data consists of K sequences, with lengths T_k, $k = 1, \ldots, K$, of the state and feature,

$$s_1^k, \ldots, s_{T_k}^k, \quad x_1^k, \ldots, x_{T_k-1}^k, \quad k = 1, \ldots, K. \tag{2}$$

In the case of missing data, we allow some states to be unknown or missing, which we denote as $s_t^k = ?$. We will assume that all of the features, however, are given, since there are many methods that can be used to fill in feature values that are missing, ranging from simple ones like replacing missing entries with an empirical average, or using the last known value, to sophisticated ones like model-based imputation [2].

2 Fitting without missing data

In this section we describe how to fit the model parameters when there is no missing data. This material is straightforward and well known; we include it for completeness, and also to refer back to when we consider the more complex case when there is missing data.

2.1 Log-likelihood

The log-likelihood of the dataset (2) is

$$L(\theta) = \sum_{k=1}^{K} \sum_{t=1}^{T_k - 1} \log(P_t^k)_{s_t^k s_{t+1}^k}, \tag{3}$$

with $P_t^k = \phi(x_t^k)$, which depends on θ. The first sum is over the K data sequences and the second sum is over $T_k - 1$ entries in each sequence. With the logistic predictor (1), the log-likelihood is

$$L(\theta) = \sum_{k=1}^{K} \sum_{t=1}^{T_k - 1} \left((y_t^k)_{s_{t+1}^k} - \log(\mathbf{1}^T \exp y_t^k) \right),$$

where $y_t^k = A_{s_t^k} x_t^k + b_{s_t^k}$. The log-likelihood is a concave function of y_t^k, and hence a concave function of θ, *i.e.*, $A_1, \ldots, A_n$ and $b_1, \ldots, b_n$. (We use the fact that the log-sum-exp function is convex, and the composition of an affine function and a convex function is convex [7,§3.2.2].)

2.2 Fitting

We choose the parameter θ to maximize the log-likelihood of the training dataset minus a regularizer function of the parameter. To choose θ we solve the optimization problem

$$\text{maximize} \quad L(\theta) - R(\theta) \tag{4}$$

with variable θ. Here the convex function $R : \Theta \to \mathbf{R}$ is a regularizer, meant to discourage over-fitting, *i.e.*, to improve the log-likelihood on new, unseen data. This fitting problem is a convex optimization problem, readily solved by a number of methods [7].

2.3 Regularizers

We describe a few useful regularizers here. Each of these includes a positive hyper-parameter λ that is selected via held-out or cross validation. Most of the regularizers that we describe below are functions only of the parameter matrices $A_1, \ldots, A_n$ and not the offset parameters $b_1, \ldots, b_n$. We will use the notation

$$A = \begin{bmatrix} A_1 \\ \vdots \\ A_n \end{bmatrix} \in \mathbf{R}^{n^2 \times m}$$

to denote the vertically stacked parameter matrices. Let $a_1, \ldots, a_m \in \mathbf{R}^{n^2}$ denote the columns of A, *i.e.*,

$$A = \begin{bmatrix} a_1 & \cdots & a_m \end{bmatrix}.$$

The vector a_j contains all the model coefficients that multiply x_j, the jth component of the feature vector.

Sum of squares. The simplest and most traditional regularizer is the sum of squares regularizer,

$$R(\theta) = \lambda \|A\|_F^2 = \lambda \sum_{i=1}^{n} \|A_i\|_F^2 = \lambda \sum_{j=1}^{m} \|a_j\|_2^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This regularizer is separable, *i.e.*, a sum of functions of each (A_i, b_i). This means the entire fitting problem is separable in (A_i, b_i), and can be solved as n separate sum of squares regularized logistic regression problems.

Sum of column norms. The sum-of-column-norms regularizer is

$$R(\theta) = \lambda(\|a_1\|_2 + \cdots + \|a_m\|_2).$$

(Note the norms here are not squared; if they were, this would coincide with the sum of squares regularizer.) This non-separable regularizer is well known to promote column sparsity, *i.e.*, $a_j = 0$ for many values of j [31]. This corresponds to feature selection, since when $a_j = 0$, the feature x_j is not used.

For λ large enough, this regularizer results in all a_j being zero, *i.e.*, a predictor that does not depend on any of the features. Indeed the threshold or critical value of λ is

$$\lambda^{\max} = \max_{j=1,\ldots,m} \|\nabla_{a_j} L(0)\|_2.$$

(This result can be derived from basic convex analysis.) The gradients are readily expressed in terms of the data, and readily computed. We have $A = 0$ if and only $\lambda \geq \lambda^{\max}$. This is useful for selecting values of λ.

Dual norm. Another interesting non-separable regularizer is the dual norm regularizer,

$$R(\theta) = \lambda \|A\|_*,$$

where $\|\cdot\|_*$ is the dual of the spectral norm, *i.e.*, the sum of the singular values. This regularizer encourages A to be low rank [23]. When A has rank r, say, $A = UV$ with $U \in \mathbf{R}^{n^2 \times r}$ and $V \in \mathbf{R}^{r \times m}$, we can decompose the prediction $\phi(x_t)$ into two steps. First we compute $z_t = Vx_t \in \mathbf{R}^r$, and then we form $Ax_t = Vz_t$. We can interpret the vector time series z_t as a latent or compressed feature vector, which contains everything we need to determine $P_t = \phi(x_t)$ [3].

As is the case with the sum of column norms regularizer, for λ large enough, the dual norm regularizer results in $A = 0$. For the dual norm regularizer the critical value of λ is

$$\lambda^{\max} = \|\nabla_A L(0)\|_2,$$

where $\|\cdot\|_2$ is the spectral norm or maximum singular value. The gradient is readily expressed in terms of the data and computed. We have $A = 0$ if and only $\lambda \geq \lambda^{\max}$. This is useful for selecting values of λ.

Laplacian across states. The next regularizer we mention is Laplacian regularization across the discrete states, which depends on the offset coefficients $b_1, \ldots, b_n$ as well as the coefficient matrices $A_1, \ldots, A_n$. This regularizer starts with a weighted graph on the states $1, \ldots, n$, given by the weighted adjacency matrix $W \in \mathbf{R}_+^{n \times n}$. These weights express our prior knowledge of similarity or closeness between the states: W_{ij} large means that states i and j are considered similar. This might arise when the states are discretizations of continuous quantities, for example, deciles. In this case we would use a chain graph to signify that state 3 (third decile) is close to states 2 and 4 (second and fourth deciles). Laplacian regularization is the quadratic form

$$R(\theta) = \lambda \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \left(\|A_i - A_j\|_F^2 + \|b_i - b_j\|_2^2 \right),$$

which encourages the parameters of the rows of ϕ associated with similar states to be near each other.

When the graph is connected and λW_{ij} are large, this regularization encourages all A_i being very close, and all b_i being close. This implies that the distributions $\phi_i(x)$ are very close. This in turn is interpretable as modeling s_t not as Markov, but as independent samples from a (feature-dependent) distribution on the states.

Laplacian across features. The final regularizer we consider is Laplacian regularization across the features. This regularizer starts with a weighted graph on the features $1, \ldots, m$, given by a weighted adjacency matrix $W \in \mathbf{R}_+^{m \times m}$. These weights express similarity between the effect the features have on the transition matrix. This might arise when the features are one-hot embeddings of an ordinal quantity, *e.g.*, age. In this case we can use a chain graph to signify that the model coefficients for age 50 should be close to the model coefficients for ages 49 and 51. Laplacian regularization is the quadratic form

$$R(\theta) = \lambda \sum_{i=1}^{m} \sum_{j=1}^{m} W_{ij} \|a_i - a_j\|_2^2,$$

where, again, a_i is the ith column of A.

Choosing one or more regularizers. The choice of one or more regularizers, as well as the associated hyper-parameters, is driven by several objectives, and of course the specific application. The first objective is good out of sample validation, as described in the next section. The second is interpretability of the resulting feature-dependent Markov chain model. For example, using a dual norm regularizer gives us z_t, which we can interpret as a latent feature; a sum of column norms regularizer gives us feature selection.

2.4 Judging a predictor

We judge a predictor ϕ by its log-likelihood (3) on (unseen) test data

$$\tilde{s}_1^k, \ldots, \tilde{s}_{\tilde{T}_k}^k, \quad \tilde{x}_1^k, \ldots, \tilde{x}_{\tilde{T}_k - 1}^k, \quad k = 1, \ldots, \tilde{K}. \tag{5}$$

When the test data contains missing data, we only include terms in the log-likelihood (3) for which both s_t^k and s_{t+1}^k are known, *i.e.*, we judge using the log likelihood over non-missing pairs

$$L^{\text{nmp}}(\theta) = \sum_{k=1}^{K} \sum_{s_t^k, s_{t+1}^k \neq ?} \log(P_t^k)_{s_t^k s_{t+1}^k}. \tag{6}$$

3 Fitting with missing data

We now consider the more complex and difficult case where some of the states s_t^k in our training dataset are unknown or missing. If there are more than just a few missing states, the log-likelihood is intractable to compute, much less optimize over, but we can fit the model by approximately maximizing the log-likelihood minus a regularization function using the well known expectation-maximization (EM) algorithm.

Without loss of generality, we assume that the first and last state in each sequence, s_1^k and s_T^k, are known. If they are not, we can truncate that sequence to the first and last known value of s_t^k. We remind the reader that we assume all the features are known, *i.e.*, none are missing.

3.1 Log-likelihood with missing data

When there is missing data, the (marginalized) log-likelihood is a complicated function of θ. It has the form

$$L(\theta) = \sum_{k=1}^{K} \log \left(\sum_{s_1^k, \ldots, s_T^k} \mathbf{Prob}(s_1^k, \ldots, s_T^k \mid \text{known}; \theta) \prod_{t=1}^{T_k - 1} (P_t^k)_{s_t^k s_{t+1}^k} \right),$$

where the first sum is over the sequences, the sum inside the log is over all possible values of the missing entries, and $\mathbf{Prob}(s_1^k, \ldots, s_T^k \mid \text{known}; \theta)$ is the conditional probability of the sequence $s_1^k, \ldots, s_T^k$ conditioned on the observed state values, under our model with parameter value θ. (The matrix P_t^k also depends on θ, but we do not explicitly show this dependence.) The inner sum is over n^M terms, where M is the number of missing state values. Unless M is very small, it is intractable to compute $L(\theta)$, or to maximize it minus a regularizer. So we turn to a powerful heuristic, EM.

3.2 Expectation-maximization

The expectation-maximization (EM) algorithm is a powerful heuristic for approximately performing maximum likelihood on incomplete data [11], and is perfectly suited for the task at hand. We adopt notation commonly used for EM, defining X to consist of all s_t^k that are known, and Z to consist of all s_t^k that are not known.

EM consists of two steps: the *E-step* and the *M-step*, which are iterated to generate a sequence of parameters θ^j, starting from an initial guess θ^0, where the superscript denotes iteration (of the EM algorithm). It is well known that each EM iteration will increase the objective, but there is no guarantee that the global maximum will be reached [30]. However, in our experiments, we have found it to be a very good heuristic. We describe the actual realization of these steps for our particular problem below.

The E-step. In the E-step, we construct an approximate log-likelihood function using θ^j,

$$Q^j(\theta) = \mathop{\mathbf{E}}_{Z|X;\theta^j} \log \mathbf{Prob}(X, Z; \theta)$$

$$
\begin{aligned}
&= \mathop{\mathbf{E}}_{Z|X;\theta^j}\left[\sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\log(P_t^k)_{s_t^k s_{t+1}^k}\right] \\
&= \sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\sum_{Z}\mathbf{Prob}(Z \mid X;\theta^j)\log(P_t^k)_{s_t^k s_{t+1}^k} \\
&= \sum_{k=1}^{K}\sum_{t=1}^{T_k-1}\sum_{s_t^k,s_{t+1}^k}\mathbf{Prob}(s_t^k, s_{t+1}^k \mid X;\theta^j)\log(P_t^k)_{s_t^k s_{t+1}^k}.
\end{aligned}
$$

Here P_t^k depends on θ, but we do not explicitly show this dependence. Computing Q^j boils down to finding the joint probabilities $\mathbf{Prob}(s_t^k, s_{t+1}^k \mid X;\theta^j)$, which are weights in the log-likelihood.

We need to compute the joint probabilities $\mathbf{Prob}(s_t^k, s_{t+1}^k \mid X;\theta^j)$ for $t = 1,\ldots,T_k-1$ and $k = 1,\ldots,K$. For each sequence $s_1^k,\ldots,s_{T_k}^k$, we do the following. (We will suppress the dependence on k, *i.e.*, we write s_t, x_t, and P_t in place of s_t^k, x_t^k, and P_t^k.) First we compute $P_1,\ldots,P_{T_k-1}$ based on θ^j. If s_t and s_{t+1} are both known, then $\mathbf{Prob}(s_t, s_{t+1} \mid X,\theta^j)$ is deterministic, *i.e.*, equal to one for the observed transition and zero for others. To find the joint distribution of successive states when either s_t or s_{t+1} is not known, we let $p \le t$ denote the time of the last previous known state, and $q \ge t$ denote the time of the next known state. Since s_p and s_q are known, we have (by the Markov property)

$$
\mathbf{Prob}(s_t, s_{t+1} \mid X;\theta^j) = \mathbf{Prob}(s_t, s_{t+1} \mid s_p, s_q;\theta^j).
$$

By Bayes rule,

$$
\begin{aligned}
\mathbf{Prob}(s_t, s_{t+1} \mid s_p, s_q;\theta^j) &\propto \mathbf{Prob}(s_p, s_q \mid s_t, s_{t+1};\theta^j)\,\mathbf{Prob}(s_t, s_{t+1};\theta^j), \\
&= \mathbf{Prob}(s_p \mid s_t;\theta^j)\,\mathbf{Prob}(s_q \mid s_{t+1};\theta^j)\,\mathbf{Prob}(s_t, s_{t+1};\theta^j), \\
&= (P_p\cdots P_{t-1})_{s_p s_t} \circ (P_{t+1}\cdots P_{q-1})_{s_{t+1}s_q} \circ (P_t)_{s_t s_{t+1}},
\end{aligned}
$$

where $\circ$ is the elementwise (Hadamard) product. This quantity is easy to compute recursively. If we have a sequence of length T with M missing entries, the complexity of evaluating the model to get $P_1,\ldots,P_{T-1}$ is Tn^2m, and the complexity of evaluating the joint probabilities above is Mn^3. Therefore, the overall complexity is $n^2(Tm+Mn)$. The E-step is very similar in nature to the forward-backward algorithm used for hidden Markov models [4].

The M-step. In the M-step, we choose θ^{j+1} as the solution of

$$
\text{maximize } Q^j(\theta) - R(\theta),
$$

where R is a convex regularizer, which is a convex optimization problem. The problem above is simply a weighted version of the fitting problem (4) with more data points, so similar methods can be used to solve it. That is, each pair s_t^k, s_{t+1}^k can result in one or more data points. When both are not missing, it results in one data point with weight one. When just one is missing, it results in n data points with weights that sum to one. When both are missing, it results in n^2 data points with weights that sum to one. In sum, if there are M missing entries in a sequence of length T, we will have at most roughly M^2+T-M pairs. In the logistic regression nomenclature, it is equivalent to a dataset with probabilistic outcomes (over $\{1,\ldots,n\}$) and weights attached to each data point. Also, we note that in the case of all known entries, the approximate log-likelihood is the same as the case without missing data, so EM will converge in just one step.

Initialization. EM requires an initial parameter θ^0. A simple initialization for EM uses the log-likelihood found using all non-missing pairs of successive states given in (6). We choose θ^0 as the maximizer of

$$L^{\mathrm{nmp}}(\theta) - R(\theta).$$

This function is concave, so this is a convex optimization problem.

4 Implementation

In this section we describe our implementation of the ideas described in this paper, which have been collated into an open-source Python package `mkvchain`, which is freely available online at

`www.github.com/cvxgrp/mkvchain`.

The core class in the package is `FeatureDependentMarkovChain`, which, at initialization, takes the number of states, and optionally a sparsity pattern represented by a binary mask, various regularization parameters, and optimization parameters such as tolerances and the maximum number of EM iterations. To fit the model, we call the `fit` method, which takes the arguments:

- `states`. A list of (integer) states. Missing entries are indicated by NaNs.
- `features`. A numpy array of features, with length at least the number of states minus one.
- `lengths`. A list of (integer) lengths of each sequence. If the data consists of only one sequence, then this argument should be equal to `[len(states)]`.

This method can also be warm-started by setting `warm_start=True`. The `fit` method implements the EM algorithm described in Sect. 3, using `pytorch` for matrix operations [21], the L-BFGS algorithm for when the regularizer is smooth, and an adaptive proximal gradient method for when the regularizer includes a non-smooth regularizer, *e.g.*, the sum of column norms or dual norm.

Once the model is fit, it can be used via either the `predict` or `score` method. The `predict` method takes as its argument a feature matrix and returns a transition matrix for each row of the feature matrix. The `score` takes the same first three arguments as the `fit` method, and evaluates the log-likelihood of the dataset using only consecutive known states, *i.e.*, (6).

5 Example: Trading volume

In this example we fit a feature-dependent Markov chain to the daily trading volume of a stock.

States. We start with the daily trading volume, in number of shares, of the JNJ (Johnson & Johnson) stock ticker, from 2000 to 2021, shown in Fig. 1. We split this data into a length 3000 training sample, a length 1000 validation sample, and a length 1309 test sample, corresponding roughly to years 2000–2011, 2012–2015, and 2015–2021, respectively.

We consider as the discrete state the quintile of the daily volume, resulting in $n = 5$ states. (The thresholds for the quintiles are determined based on the training data.) We label these states as low, below average, average, above average, and high volume. The quintile thresholds are given in Fig. 1 as horizontal dashed lines, including the 0th and 100th percentiles (*i.e.*, the minimum and maximum values). We can see one period in 2012 when the trading volume exceeds the maximum value that occurred in the training data.

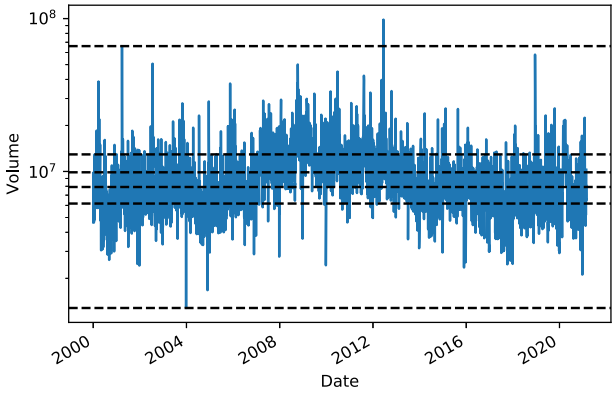

Fig. 1 Daily log volume of JNJ over the sample period. The dashed horizontal lines show quintile boundaries, determined by the training data

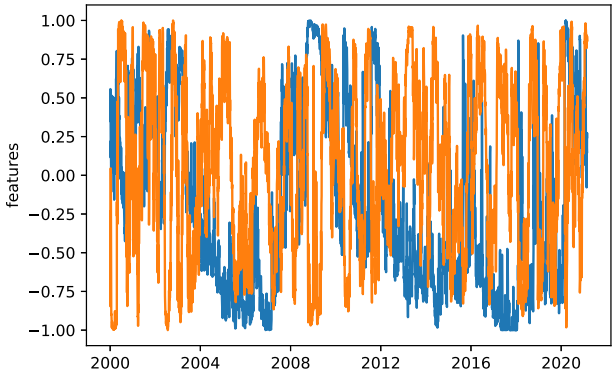

Fig. 2 Preprocessed VIX and RET60D over the sample period

Features. We consider as features the previous close of the CBOE volatility index (VIX), which is a measure of implied future volatility, and the trailing 60-day average return of JNJ. (The total number of features is $m = 2$.) We preprocess the features by performing a quantile transform so that each feature has a marginal uniform distribution over the interval $[-1, 1]$, based on the distribution of values in the training set. These features through the sample period are plotted in Fig. 2.

5.1 Fitting without missing data

We fit four models to the (full) original data, which are described in order below.

- *Empirical.* The empirical transition matrix.
- *Constant Lap-reg.* A constant transition matrix (*i.e.*, $A = 0$), with chain graph Laplacian regularization on the states, tuned using the validation sample.
- *Feature-dependent.* A feature-dependent model with sum of squares regularization, tuned using the validation sample.
- *Feature-dependent Lap-reg.* A feature-dependent model with sum of squares plus chain graph Laplacian regularization on the states, both tuned using the validation sample.

Table 1 Results without missing data. The first row gives the log-likelihood and MAE of the baseline empirical model on train, validation, and test samples. The remaning three rows give the increase in log-likelihood over the baseline model

Predictor	Train	Validation	Test	Test MAE
Empirical baseline	−3900.401	−1371.396	−1691.016	0.1679
Constant Lap-reg	−0.17	0.07	0.88	0.1680
Feature-dependent	72.47	19.22	21.77	0.1627
Feature-dependent Lap-reg	71.18	19.92	28.77	0.1624

Log-likelihood results. In Table 1 we give the train, validation, and test log-likelihoods of the four models (the final column, test MAE, will be explained below). The first row gives the log-likelihood for the simple empirical model, and the remaining three rows give the log-likelihood minus the value for the empirical model. (Thus we use the empirical model as our reference or baseline; positive numbers means a lift in log-likelihood over the baseline empirical model.) The best model, judged by test log-likelihood, is the feature-dependent model with Laplacian regularization on the states. We see that adding Laplacian regularization to the constant model did not provide a significant lift in log-likelihood.

A next day volume predictor. As a further check on our feature-dependent Markov chain models, we use them to create a predictor of the next day volume quantile, a number between 0 and 1. We do this by mapping the distribution $\phi_i(x_t)$ (for $s_t = i$) into its median value $\hat{q}_{t+1}$, interpreting $\phi_i(x_t)$ as a density on [0, 1] with constant value $(\phi_i(x_t))_j$ on the interval $((j-1)/5, j/5]$, $j = 1, \ldots, 5$. We compare this prediction to q_{t+1}, the actual quantile of volume on day t. In the last column of Table 1, we give the mean absolute error (MAE) for some of our models, where lower is better. We see that performance at this task is reasonably consistent with the test log-likelihood values in Table 1.

Of course many other methods could be used to more directly predict the next day quantile, given the features. We show this approach only to verify that our feature dependent Markov models improve the prediction over the empirical model.

5.2 Fitting with missing data

The original data is full, *i.e.*, has no missing data. But we artificially introduce missing data into the training sample by setting 1500 (about half) of the volume states to ? at random. In Fig. 3 we show the histogram of the length of consecutive missing entries in the training dataset. For example, there are 745 consecutive pairs where both states are known, and the longest sequence of consecutive missing entries is 6.

Models. We fit six models, which are described in order below.

- *Constant.* A constant baseline model (*i.e.*, $A = 0$). We ignore pairs of states with one or more missing values. This is the empirical transition matrix based only on the known pairs of consecutive states.
- *Constant EM.* A constant model. We use EM to deal with missing states.
- *Constant EM Lap-reg.* A constant model with chain graph Laplacian regularization on the states, tuned using the validation sample. We use EM to deal with missing states.
- *Feature-dependent.* A feature-dependent model with sum of squares regularization, tuned using the validation set. We ignore pairs of states with one or more missing values.

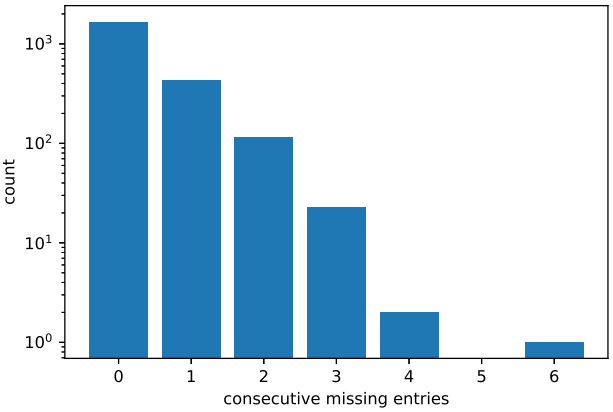

Fig. 3 Histogram of consecutive missing entries in the training dataset

Table 2 Results with missing data. The first row gives the log-likelihood and MAE of the baseline empirical model on train, validation, and test samples. The remaining five rows give the increase in log-likelihood over the baseline model

Predictor	Train	Validation	Test	Test MAE
Constant baseline	−2147.901	−1382.040	−1692.203	0.1671
Constant EM	−2.27	6.01	−4.94	0.1670
Constant EM Lap-Reg	−3.04	8.45	3.65	0.1677
Feature-dependent	49.42	24.67	24.13	0.1603
Feature-dependent EM	44.87	29.03	18.50	0.1607
Feature-dependent EM Lap-reg	40.91	32.86	32.98	0.1611

- *Feature-dependent EM.* A feature-dependent model with sum of squares regularization, tuned using the validation set. We use EM to deal with missing states.
- *Feature-dependent EM Lap-reg.* A feature-dependent model with sum of squares regularization plus chain graph Laplacian regularization on the states, both tuned using the validation sample. We use EM to deal with missing states.

Log-likelihood results. In Table 2 we give the train, validation, and test log-likelihood (evaluated using only the known pairs) of the six models. Several interesting conclusions can be drawn from the results. First, Constant EM does worse than just Constant on the known pairs in the training set, which makes sense, because it has a different objective. The feature-dependent model without EM did about as well on the test set as Constant EM, but the best model by far was feature-dependent EM with Laplacian regularization, which incorporates the features, gracefully handles the missing data, and uses the fact that adjacent states are similar. Counterintuitively, the best model in the missing data case does better than the best model fit to the full dataset. The only explanation we have is that some form of implicit regularization is going on, or that the result is just due to chance.

Next day volume prediction. Similar to the case without missing data, we computed the test MAE of the six models on the task of predicting the next day volume quantile. This is given in the last column of Table 2. The best model under this metric is Feature-dependent without EM, instead of Feature-dependent EM Lap-reg, which has the best test log-likelihood.

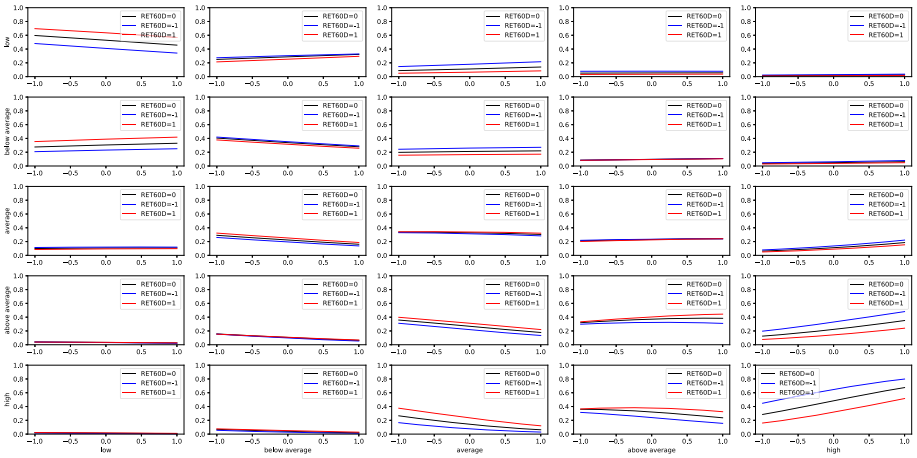

Fig. 4 Transition matrix entries versus VIX, for RET60D at its minimum, median, and maximum value.

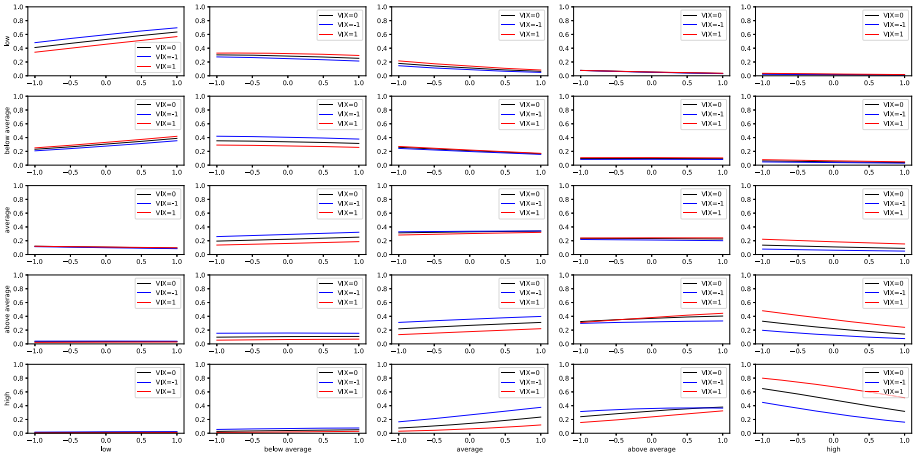

Fig. 5 Transition matrix entries versus RET60D, for VIX at its minimum, median, and maximum value.

5.3 Interpretations

In Fig. 4 we show the entries of the transition matrix of the Feature-dependent EM Lap-reg model versus VIX, for RET60D at its minimum, median, and maximum value. As expected, as VIX becomes larger, so does the probability that the volume increases. In Fig. 5 we show the entries of the transition matrix versus RET60D, for VIX at its minimum, median, and maximum value. The return has a more complicated effect than VIX; as the return becomes larger, the probability that JNJ stays in the high volume state becomes lower. Also, as the return becomes larger, the probability that the volume stays at average becomes larger. Finally, in Fig. 6 we show a heatmap of the entry P_{55} versus all possible values of RET60D and VIX. This heatmap shows that the entries have a nontrivial dependence on the features.

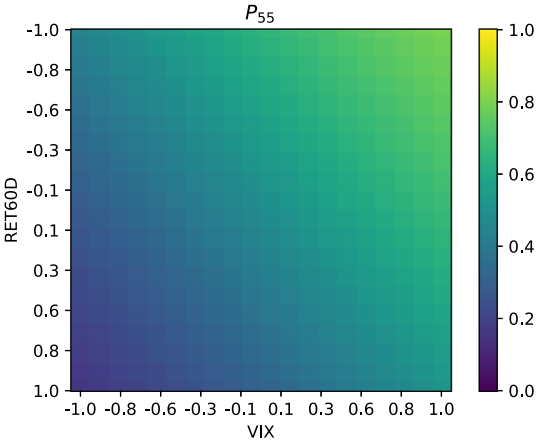

Fig. 6 A heatmap of the entry P_{55} versus RET60D and VIX.

6 Example: Baseball

In this section we fit a feature-dependent Markov chain to the game state in baseball.

Data. We gather data representing all 191051 at-bats that occurred in the 2018 MLB baseball season from the data provider Retrosheet [24]. For each at-bat, the data has various attributes, such as the pitcher, the batter, the score of the baseball game, and the runner positions. We also gather, separately for each pitcher and batter, various aggregate statistics from the 2017 season, *e.g.*, their batting average, strike out percentage, single percentage, as well as their handedness. We perform a temporal 60/20/20 split of the season into a training, validation, and test sample. None of the states are missing. Unlike the volume example in Sect. 5 where we had a single state sequence (*i.e.*, $K = 1$), in this example we have $K = 43627$ state sequences, each of which constitutes a half-inning of baseball.

State. We consider as the discrete state the number of outs (0, 1, or 2) and whether or not there is a runner on first, second, and third base during an at-bat, expressed as three values in $\{0, 1\}$. We also add a state for inning over, which is effectively a three-out state. Thus, the total number of states is $n = 3 \times 2 \times 2 \times 2 + 1 = 25$. Of course, the inning over state is absorbing. Due to the rules of baseball, only some of the state transitions are possible. For example, outs cannot decrease, and the total number of runners on base cannot increase by more than one. We map out 294 of the $n^2 = 625$ transitions that are possible according to the rules of baseball and mask the transition matrix accordingly. (Only 279 of these 294 transitions occur in the 2018 season.) For future reference, the state (a, b, c, d) means a outs, $b = 0$ means no runner on first base, and $b = 1$ means a runner is on first base (c and d have the same meaning as b, but for second and third base). The inning over state is denoted $(3, 0, 0, 0)$.

Features. We consider as features the one-hot version of the current run difference (clipped to $[-3, 3]$), whether the batting team is home or away, the one-hot inning, the one-hot lineup position, the handedness of the pitcher and batter, including whether they have the same dominant hand, and all of the aggregate at-bat result statistics described above. The total number of features is $m = 64$. With these features, our logistic transition model is *additive* in the pitcher and batter features, *i.e.*, there is no interaction between the pitcher and better features, except for their handedness. This is a reasonable assumption for our simple example, but in practice one would probably need to add pitcher-batter interactions. We standardize the

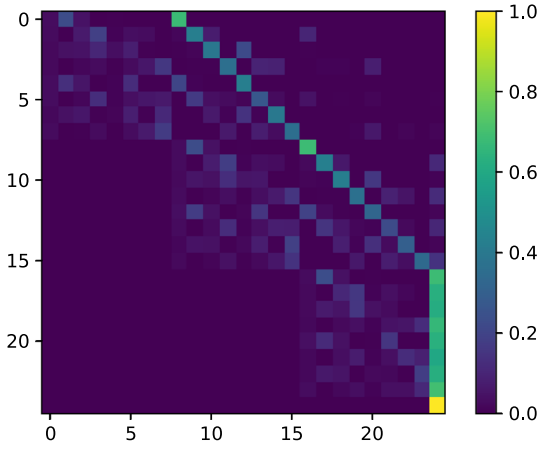

Fig. 7 A heatmap of the empirical transition matrix

features so that they have zero mean and unit standard deviation on the training dataset. Since some of the pitchers and batters did not play in the 2017 season, we set their features to NaN, and ignore transitions with any feature vectors that contain NaNs in fitting and evaulating all of our models.

Empirical model. We begin by considering the empirical transition matrix between the 25 states, which is plotted in Fig. 7 for the training set. All of these transitions are one of the 294 transitions we mapped out as possible. The first 8 states correspond to 0 outs, the next 8 correspond to 1 out, and the final 8 (except the last) correspond to 2 outs. Thus, it makes sense that the strictly lower triangular (block) entries are all zero, since outs cannot decrease. Also, there are only a few transitions from 0 outs to 2 outs, which can only happen when there is at least one runner on base. The last row corresponds to the (3, 0, 0, 0) state, which is absorbing. The transition matrix is auditable; we can easily inspect it to make sure it makes sense. For example, the most probable transition from (0, 0, 0, 0) is to (1, 0, 0, 0), which means the batter got out. The probability of this transition is 68.3%, meaning the probability a batter does not get out with the bases empty is 31.5%, which corresponds almost exactly with the overall published league on base percentage (OBP) of 31.8%. As another example, the second most probable transition for (1, 0, 0, 1) is to (2, 0, 0, 0), which means the runner on third scored, but the batter got out, most likely indicating a sacrifice fly or similar play. This transition occurs with probability 20%.

Models. We fit four models, which are described in order below.

- *Empirical.* The empirical transition matrix.
- *Feature-dependent with sum of squares regularization.* A feature-dependent Markov chain model with sum of square regularization, tuned using the validation sample.
- *Feature-dependent with column norm regularization.* A feature-dependent Markov chain model with column norm regularization, chosen so that 9 of the 64 features were used. (These features are listed below.)
- *Feature-dependent subset features.* A feature-dependent Markov chain model using the 9 features selected by the solution to the column norm problem, with sum of squares regularization, tuned using the validation sample.

Results. In Table 3 we give the average test log-likelihood of the four models. The feature-dependent sum of squares model and feature-dependent subset features model were tied for the best test log-likelihood. However, the latter model uses significantly fewer features

Table 3 The first row gives the log-likelihood of the baseline empirical model on train, validation, and test samples. The remaning three rows give the increase in log-likelihood over the baseline model

Predictor	Train	Validation	Test
Empirical baseline	−147840.7744	−15071.1119	−14487.9775
Feature-dependent sum of squares	2383.98	58.11	75.62
Feature-dependent column norm	251.90	16.91	24.82
Feature-dependent subset features	893.40	60.14	74.48

(9 versus 64), and so might be preferred. Also, it is worth noting that when we re-fit the model with the features that had nonzero coefficients in the column norm model, the test log-likelihood significantly improved.

Interpretations. Since the features are standardized, we can look at the model coefficients and find which features have the biggest effect on certain transitions. Consider the transition from $(0, 0, 0, 0)$ to $(0, 0, 0, 0)$, which occurs when the batter gets a home run. Some of the features that have the most positive effect on this transition are whether the batter is in the fourth lineup spot, the fraction of the time that the pitcher gives up home runs, RBIs, and walks. The features that affect this transition the least are whether the batter is in the eighth or ninth lineup spot, whether the game is in extra innings, and the fraction of the time a pitcher gives up triples.

Feature selection. In the column norm-regularized model, the selected features were whether the batter is first or last in the lineup, whether the batting team is down by 3 or more runs, the fraction of the time a batter hits a triple, whether it is the ninth inning, and the fraction of the time the pitcher gets a home run, walk, and strike. At least qualitatively, these all seem like very important features for predicting the probability of different transitions between game states.

Fictitious game simulations. As a further check on our models, we can run full baseball game simulations and check the results. Here we perform fictitious simulation, where we always have the same pitcher and the same batter for each respective team for the entire game. Of course, the very same model could be used for simulating a real game, including the real batter order, pitching substitutions, and so on, but for simplicity we fix the pitcher and batter. We consider a game between the SF Giants and the Seattle Mariners, played at the Giants' stadium. For the Mariners, we have Marco Gonzales pitching and David Frietas batting. For the Giants, we have Tyson Blach pitching and Buster Posey batting. We ran ten thousand full game simulations using the feature-dependent with sum of squares model, and calculated the final runs scored by each team in both simulations. The win probability under this model for the Giants was 54.3%, and on average they led by 0.25 runs at the end of the game. (If we ran the game simulations using the constant model, the win probability would be 50%, since the model is symmetric and has identical run distributions for each side.) The histogram of the Giants' runs minus the Mariners' runs over all the simulations is plotted in Fig. 8.

Even though this simulation with a fixed pitcher and batter for each team is unrealistic, it is still interesting and informative. For example, if we switch these to Felix Hernandez pitching and Mitchell Haniger batting for the Mariners, and Johnny Cueto pitching and Gorkys Hernandez batting for the Giants, the Giants win probability drops to 38.51%, the average final lead is −0.937, and the lead histogram is shown in Fig. 9.

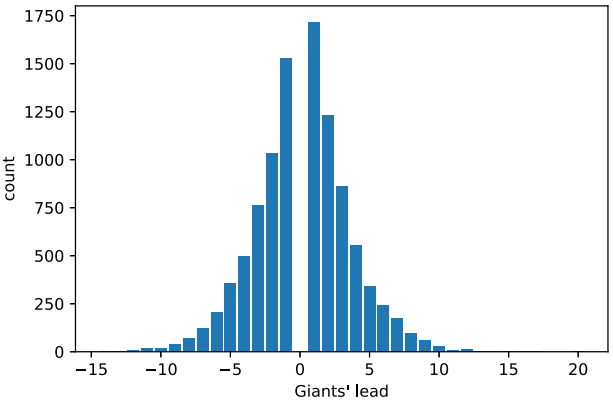

Fig. 8 A histogram of the Giants' lead over the Mariners over the ten thousand game simulations, with Marco Gonzales pitching and David Freitas batting for the Mariners, and Tyson Blach pitching and Buster Posey batting for the Giants

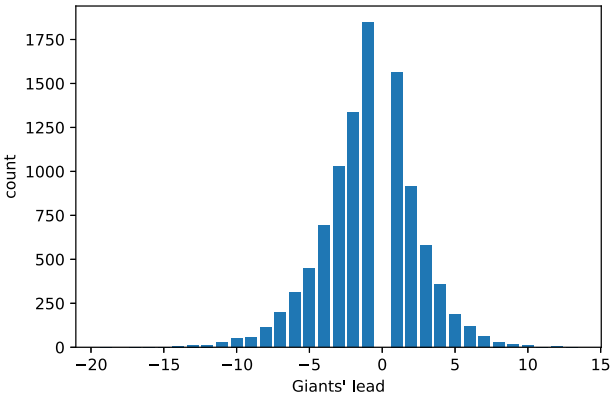

Fig. 9 A histogram of the Giants' lead over the Mariners over the ten thousand game simulations, with Felix Hernandez pitching and Mitchell Haniger batting for the Mariners, and Johnny Cueto pitching and Gorkys Hernandez batting for the Giants

7 Previous work

Markov chains. The topic of Markov chains or processes has a long history, with many books written on the subject, *e.g.*, [15, 19, 25, 28]. Markov chains and the Markov property are both fundamental to stochastic control [6]. The Markov chain was also the basis for PageRank, which formed the initial version of the search website Google [20].

Hidden Markov models. Hidden Markov models (HMMs) are Markov chains where the state is unobservable, and instead, evaluations of a random function of the state are observed [22]. As mentioned above, the forward-backward algorithm that is commonly used to fit parameters in a HMM is very similar in nature to the E-step of the EM algorithm described in Sect. 3. The authors are unaware of HMM variants where the hidden state transition probabilities depend on covariates.

Hazard models. Many other widely used models are specific instances of the model we described in this paper. One example is the proportional hazards model from statistics. In

a proportional hazards model, there is a Markov chain with two states: the hazard has not yet happened, and the hazard has happened, which is an absorbing state. The discrete-time version of the Cox proportional hazards model [9], which models the hazard likelihood using a logistic function, is exactly our model. The original application of the Cox proportional hazards model was to feature-dependent life tables [9]; since then it has been applied to many areas, including modeling bank failure [17], system reliability [18], and terrorism assessment [29].

EM for fitting Markov chains. Fitting a Markov chain to data is often performed by simply computing the empirical transition matrix. However, when there is missing data, this simple method has to discard any state pairs where one or more of the states are missing. As a solution for this issue, in 1995, Sherlaw derived the EM algorithm for the problem of fitting a Markov chain transition matrix to data with missing entries [26]. The algorithm he proposed is identical to the one we describe here when there are no features, *i.e.*, $m = 0$. Indeed, the constant EM model described in Sect. 5 fits the model in the exact same way as [26]. We note that the basic EM algorithm described by Sherlaw has been extended to more complex patterns of incompleteness, *e.g.*, the state being known to be in a subset of the possible states [10]. The methods we describe in this paper can be extended in similar ways.

Feature-dependent Markov chains. The idea of making the transition matrix of a Markov chain depend on external features is by no means new. In 1976, Boyle proposed using linear discriminant analysis (LDA) for parametrizing the transition matrix [8]. Shortly after, in 1979, Korn and Whittemore proposed a logistic model for the transition probabilities in a two-state (binary) Markov chain, and applied it to panel data related to air pollution [16]. The incorporation of covariates in a continuous-time Markov model was introduced by Kalbfleisch and Lawless in [14].

Since then, feature-dependent Markov chains have found much use in medical decision making [5, 27], among other applications. One prototypical use is to make the probability of death or the progression of particular ailments depend on age. However, the effect of these features on the transition probabilities are often derived or estimated by hand. This is hard to do with more than a few features or states. Since then, practitioners have developed what are called cohort state-transition models or Markov cohort models, which effectively fit a separate logistic regression model for each row of the transition matrix [1]. These models have been implemented, without support for missing data, as part of the R package `hesim` [13].

Contribution. Given the vast extent of the previous work, we discuss our contribution to the literature here. Our work can be viewed as a applying the Markov chain EM algorithm of Sherlaw [26] to the case where the transition matrix depends on features [16]. We give a succinct and comprehensive description of the problem, its properties, and an EM-based solution method, paired with an efficient and widely available implementation. We apply the methods to predicting changes in stock volume and baseball game state transitions, and find the method, in our experiments, to work well.

Acknowledgements The authors gratefully acknowledge conversations and discussions about some of the material in this paper with Trevor Hastie, Emmanuel Candes, Scott Harris, and Paul Bien.

References

1. Alarid-Escudero, F., Krijkamp, E., Enns, E., Yang, A., Hunink, M., Pechlivanoglou, P., Jalal, H.: Cohort state-transition models in R: A tutorial. arXiv preprint arXiv:2001.07824, (2020)

2. Allison, P.: Missing Data. Sage Publications, New York (2001)
3. Barratt, S., Dong, Y., Boyd, S.: Low rank forecasting. arXiv preprint arXiv:2101.12414, (2021)
4. Baum, L.: An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. Inequalities **3**(1), 1–8 (1972)
5. Beck, R., Pauker, S.: The Markov process in medical prognosis. Med. Decis. Making **3**(4), 419–458 (1983)
6. Bellman, R.: Dynamic programming. Science **153**(3731), 34–37 (1966)
7. Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)
8. Boyle, B.: Estimation of feature-dependent Markov process transition probability matrices. Inf. Control **32**(4), 379–384 (1976)
9. Cox, D.: Regression models and life-tables. J. Roy. Stat. Soc.: Ser. B (Methodol). **34**(2), 187–202 (1972)
10. Deltour, I., Richardson, S., Le Hesran, J.-Y.: Stochastic algorithms for Markov models estimation with intermittent missing data. Biometrics **55**(2), 565–573 (1999)
11. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc.: Ser. B (Methodol). **39**(1), 1–22 (1977)
12. Hastie, T., Tibshirani, R., Friedman, J.: The Elements Of Statistical Learning: Data Mining, Inference, And Prediction. Springer Science & Business Media, Germany (2009)
13. Incerti, D., Jansen, J.: hesim: Health Economic Simulation Modeling and Decision Analysis, (2021). R package version 0.5.0
14. Kalbfleisch, J., Lawless, J.: The analysis of panel data under a Markov assumption. J. Am. Stat. Assoc. **80**(392), 863–871 (1985)
15. Kemeny, J., Snell, L.: Markov Chains, vol. 6. Springer, New York (1976)
16. Korn, E., Whittemore, A.: Methods for analyzing panel studies of acute health effects of air pollution. Biometrics, 795–802, (1979)
17. Lane, W., Looney, S., Wansley, J.: An application of the Cox proportional hazards model to bank failure. Journal of Banking & Finance **10**(4), 511–531 (1986)
18. Makis, V., Jardine, A.: Optimal replacement in the proportional hazards model. INFOR: Inf. Sys. Oper. Res. **30**(1), 172–183 (1992)
19. Norris, J.: Markov Chains. Cambridge University Press, Cambridge (1998)
20. Page, L., Brin, S., Motwani, R., Winograd, T.: The Pagerank Citation Ranking: Bringing Order To The Web. Technical report, Stanford InfoLab, Netherlands (1999)
21. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems, pages 8024–8035, (2019)
22. Rabiner, L., Juang, B.: An introduction to hidden Markov models. IEEE ASSP Mag. **3**(1), 4–16 (1986)
23. Recht, B., Fazel, M., Parrilo, P.: Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. SIAM Rev. **52**(3), 471–501 (2010)
24. Inc. Retrosheet. Retrosheet. https://retrosheet.org/
25. Revuz, D.: Markov Chains. Elsevier, Amsterdam (2008)
26. Sherlaw-Johnson, C., Gallivan, S., Burridge, J.: Estimating a Markov transition matrix from observational data. J. Oper. Res. Soc. **46**(3), 405–410 (1995)
27. Sonnenberg, F., Beck, R.: Markov models in medical decision making: a practical guide. Med. Decis. Making **13**(4), 322–338 (1993)
28. Walrand, J.: Probability In Electrical Engineering And Computer Science: An Application-driven Course. Quorum Books, Santa Barbara, California (2014)
29. Woo, G.: Quantitative terrorism risk assessment. The Journal of Risk Finance, (2002)
30. Wu, Jeff: On the convergence properties of the EM algorithm. The Annals of Statistics, pages 95–103, (1983)
31. Yuan, M., Lin, Y.: Model selection and estimation in regression with grouped variables. J. R. Stat. Soc.: Ser. B (Statistical Methodology) **68**(1), 49–67 (2006)