Expert Systems With Applications 150 (2020) 113307

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Learning hidden Markov models with persistent states by penalizing jumps

Peter Nystrup[a,b,*], Erik Lindström[a], Henrik Madsen[b]

[a] Centre for Mathematical Sciences, Lund University, Box 118, 221 00 Lund, Sweden
[b] Department of Applied Mathematics and Computer Science, Technical University of Denmark, 2800 Kgs. Lyngby, Denmark

A R T I C L E I N F O

Article history:
Received 13 August 2019
Revised 22 January 2020
Accepted 11 February 2020
Available online 13 February 2020

Keywords:
Time series analysis
Clustering
Unsupervised learning
Regime switching
Regularization
Dynamic programming

A B S T R A C T

Hidden Markov models are applied in many expert and intelligent systems to detect an underlying sequence of persistent states. When the model is misspecified or misestimated, however, it often leads to unrealistically rapid switching dynamics. To address this issue, we propose a novel estimation approach based on clustering temporal features while penalizing jumps. We compare the approach to spectral clustering and the standard approach of maximizing the likelihood function in an extensive simulation study and an application to financial data. The advantages of the proposed jump estimator include that it learns the hidden state sequence and model parameters simultaneously and faster while providing control over the transition rate, it is less sensitive to initialization, it performs better when the number of states increases, and it is robust to misspecified conditional distributions. The value of estimating the true persistence of the state process is illustrated through a simple trading strategy where improved estimates result in much lower transaction costs. Robustness is particularly critical when the model is part of a system used in production. Therefore, our proposed estimator significantly improves the potential for using hidden Markov models in practical applications.

1. Introduction

A Markov process is a random process in which the future is independent of the past, given the present. When the state space is discrete, Markov processes are known as Markov chains. In a hidden Markov model (HMM), the probability distribution that generates an observation depends on the state of an unobserved Markov chain. The model is popular in many areas with applications ranging from face recognition (Nefian & Hayes, 1998) and natural language processing (Gales & Young, 2008; Kang, Ahn, & Lee, 2018) to wind- and solar-power forecasting (Bhardwaj et al., 2013; Pinson et al., 2008), fraud detection (Robinson & Aria, 2018), risk and return modeling in finance and energy markets (Dias & Ramos, 2014; Petropoulos, Chatzis, & Xanthopoulos, 2016; Rydén, Teräsvirta, & Åsbrink, 1998), and genome annotation in computational biology (Choo, Tong, & Zhang, 2004).

“All models are wrong but some are useful” (Box, 1979).

In many applications the model is only successful if the state sequence has a certain level of persistence. If the model is misspecified, for example because of the Markov assumption or erroneous conditional distributions, or misestimated, for example due to limited data being available, high dimensionality, or nonstationarity, it typically leads to unstable and impersistent estimates of the underlying state sequence and creation of unnecessary extra states (Bulla, 2011; Fiecas, Franke, von Sachs, & Kamgaing, 2017; Johnson & Willsky, 2013). When the state sequence contains far too many jumps, the model essentially converges to a mixture model with no temporal information, which is of limited use.

Several studies have shown the potential benefit from changing asset allocation depending on regimes inferred from financial returns (see, e.g., Bae, Kim, & Mulvey, 2014; Bulla, Mergner, Bulla, Sesboüé, & Chesneau, 2011; Guidolin & Timmermann, 2007; Nystrup, Boyd, Lindström, & Madsen, 2019; Nystrup, Hansen, Larsen, Madsen, & Lindström, 2017a; Nystrup, Hansen, Madsen, & Lindström, 2015a; Nystrup, Madsen, & Lindström, 2018); however, if the inferred regime changes too often, this results in excessive trading costs and inferior performance. When forecasting up and down regulation in power balancing markets, in addition to significant costs, there are physical constraints on how fast production can be changed in response to a change in the forecasted direction (see, e.g., Blanco & Morales, 2017; Nielsen, Morales, Zugno, Pedersen, & Madsen, 2016). Hence, it is of paramount importance to develop an estimation framework where expert knowledge can be utilized to estimate persistent models.

* Corresponding author at: Centre for Mathematical Sciences, Lund University, Box 118, Lund 221 00, Sweden.

E-mail addresses: peter.nystrup@matstat.lu.se (P. Nystrup), erik.lindstrom@matstat.lu.se (E. Lindström), hmad@dtu.dk (H. Madsen).

https://doi.org/10.1016/j.eswa.2020.113307
0957-4174/© 2020 Elsevier Ltd. All rights reserved.

1.1. Related work

Given a long sequence of time series data, it is often desirable to partition it into segments, where each segment can be explained by as simple a model as possible. Variants of the problem have been studied in several contexts, including change-point detection (Gustafsson, 2000; Nystrup, Hansen, Madsen, & Lindström, 2016; Oh & Han, 2000), segmentation (Hallac, Nystrup, & Boyd, 2019; Katz & Crammer, 2015), trend filtering (Kim, Koh, Boyd, & Gorinevsky, 2009), and mixture models (Dias, Vermunt, & Ramos, 2015; Samé, Chamroukhi, Govaert, & Aknin, 2011; Verbeek, Vlassis, & Kröse, 2003).

The task of segmenting a time series is different from clustering in general, because the time ordering of the data must be taken into account. The different methods make different assumptions about the data. The ergodic HMM assumes that the underlying states repeat themselves, with some structure to when transitions are likely to occur. It is possible, though, to impose additional constraints to ensure nonrepeatability of the states, in which case the model is referred to as a left-to-right HMM (Bakis, 1976).

The standard, two-state Gaussian HMM has been extended in numerous ways to overcome its shortcomings in specific applications. Through conditional distributions with heavier tails (Bulla, 2011) or time-varying parameters (Nystrup, Madsen, & Lindström, 2017b) it is possible to increase persistence and improve the model's ability to reproduce long memory. Other studies have relaxed the Markov assumption on sojourn-time distributions (Bulla & Bulla, 2006; Johnson & Willsky, 2013; Langrock & Zucchini, 2011), considered higher-order Markov chains (Petropoulos, Chatzis, & Xanthopoulos, 2017; Shamshad, Bawadi, Hussin, Majid, & Sanusi, 2005), and increased the size of the hidden state space (Beal, Ghahramani, & Rasmussen, 2002; Ghahramani & Jordan, 1997; Nystrup, Madsen, & Lindström, 2015b; Sedoc, Rodu, Foster, & Ungar, 2018). In a Bayesian framework the level of persistence can be adjusted through a hyperparameter in the hierarchical Dirichlet prior distribution imposed on the transition probability matrix (Beal et al., 2002; Fox, Sudderth, Jordan, & Willsky, 2011).

Traditional approaches to estimating HMMs include direct numerical maximization of the likelihood function, the Expectation–Maximization (EM) algorithm (Baum, Petrie, Soules, & Weiss, 1970; Dempster, Laird, & Rubin, 1977), and Bayesian estimation using Markov chain Monte Carlo methods (Rydén, 2008). From a theoretical point of view the maximum likelihood estimate (MLE) has favorable properties; for example, it is consistent and asymptotically efficient. However, computationally it is not very tractable (Andersson, Rydén, & Johansson, 2003). This led Hsu, Kakade, and Zhang (2012) to propose a spectral algorithm for learning discrete-output HMMs based on a singular value decomposition of a matrix of joint probabilities of past and future observations, closely related to subspace identification methods used in control theory. Song, Boots, Siddiqi, Gordon, and Smola (2010) proposed a kernel-based approach for learning HMMs based on Hilbert space embedding.

Rousseeuw, Émilie Poisson Caillault, Lefebvre, and Hamad (2015) used the spectral clustering algorithm of Ng, Jordan, and Weiss (2002) to estimate the parameters of a discrete-output HMM. They were able to recover the hidden partition without prior knowledge of the specific model parameters. Subsequently, Zheng, Li, and Xu (2019) used spectral clustering to learn a continuous-output HMM based on simple features constructed from the observed time series. Their results showed that spectral clustering is more robust and more accurate than maximum likelihood estimation when the state process is less persistent. Meanwhile, both methods had issues with the highly persistent processes that are characteristic for many financial time series (Cont, 2001; Nystrup et al., 2017b; Rydén et al., 1998).

1.2. Contribution

We propose a novel way of learning HMMs based on temporal features and penalizing jumps. Penalizing jumps is key to successfully applying simple clustering techniques to learn HMMs with persistent states from temporal features. We rely on the same set of features considered by Zheng et al. (2019) combined with a variant of the framework proposed by Bemporad, Breschi, Piga, and Boyd (2018) for fitting jump models. Our contribution is to combine these two approaches into a robust estimator with possibility for utilizing prior knowledge to estimate persistent models. Without regularization on jumps it nests the standard K-means clustering algorithm of Lloyd (1982). When penalizing jumps it is reminiscent of the Viterbi Path Counting algorithm proposed by Davis and Lovell (2004). By extending several other methods, it has a wide application within expert and intelligent systems.

We extend the simulation study of Zheng et al. (2019) by considering multiple simulation lengths, different levels of persistence in the state process, models with multiple states, and the impact of misspecified conditional distributions. Our proposal of learning HMMs by clustering temporal features while penalizing jumps compares favorably to both spectral clustering and the traditional approach of maximizing the likelihood function using the EM algorithm.

The jump penalty regularizes the estimation problem and provides control over the transition rate. Other advantages of the proposed estimator include that it learns the hidden state sequence and model parameters simultaneously, it needs much fewer iterations to converge, it is less sensitive to initialization, it performs better when the number of states increases, and it is robust to misspecified conditional distributions. Robustness is particularly critical when the model is part of an expert system used in production. Therefore, our proposed estimator significantly improves the potential for using hidden Markov models in practical applications.

We illustrate the value of estimating the true persistence of the state process through a long–short trading strategy using both simulated and real financial data. The ability to control the transition rate, and through that the turnover, results in much lower transaction costs.

We begin in Section 2 by revisiting the spectral clustering approach to HMM estimation. In Section 3, we outline our proposal of learning HMMs with persistent states by clustering temporal features while penalizing jumps. The simulation study is presented in Section 4. In Section 5, the estimated states form the basis of a trading strategy that illustrates the value of estimating the true persistence of the state process. Finally, Section 6 concludes.

2. Spectral clustering

Spectral clustering of affinity graphs is a popular tool in exploratory data analysis and has enjoyed much empirical success in machine learning (Jia, Ding, Xu, & Nie, 2013; von Luxburg, 2007; Ng et al., 2002; Shi & Malik, 2000). The idea of spectral clustering is based on spectral graph theory. It treats the data clustering problem as a graph partitioning problem and constructs an undirected weighted graph with each point in the dataset being a vertex and the similarity value between any two points being the weight of the edge connecting the two vertices (Jia et al., 2013). This is commonly done by constructing a low-dimensional embedding of the affinity matrix, followed by K-means clustering in the low-dimensional space.

Algorithm 1 summarizes in six steps the spectral clustering algorithm for learning HMMs proposed by Zheng et al. (2019). The inputs are the time series $y = \{y_1, \ldots, y_T\}$ and the number of latent states K. First, a set of features is constructed from the time series. Second, an affinity matrix $A \in \mathbb{R}_{++}^{T\times T}$ is constructed from the

Algorithm 1 Spectral clustering for HMMs.

```
Input: Time series $y = \{y_1, \dots, y_T\}$ and number of latent states $K$.
1. Construct a set of standardized features from the time series $y$.
2. Form an affinity matrix $A \in \mathbb{R}_{++}^{T\times T}$ from the standardized features.
3. Compute the random-walk normalized Laplacian $L_{\text{rw}} = I - D^{-1}A \in \mathbb{R}_{++}^{T\times T}$.
4. Find the first $K$ eigenvectors of $L_{\text{rw}}$ and collect them in a matrix $U \in \mathbb{R}^{T\times K}$.
5. Group the rows of $U$ into $K$ clusters using the $K$-means algorithm.
6. Compute the transition probabilities and distributional parameters for each cluster.
Output: HMM parameters and prediction of latent states.
```

Algorithm 2 Features for HMM estimation.

```
Input: Time series $y = \{y_1, \dots, y_T\}$ and window length $l$.
1. Observation: $y_t$
2. Left absolute change: $|y_t - y_{t-1}|$
3. Right absolute change: $|y_{t+1} - y_t|$
4. Centered local mean: $\text{mean}\left[y_{t-\frac{l-1}{2}}, \dots, y_t, \dots, y_{t+\frac{l-1}{2}}\right]$
5. Centered local standard deviation: $\text{std}\left[y_{t-\frac{l-1}{2}}, \dots, y_t, \dots, y_{t+\frac{l-1}{2}}\right]$
6. Left local mean: $\text{mean}\left[y_{t-\frac{l-1}{2}}, \dots, y_t\right]$
7. Left local standard deviation: $\text{std}\left[y_{t-\frac{l-1}{2}}, \dots, y_t\right]$
8. Right local mean: $\text{mean}\left[y_t, \dots, y_{t+\frac{l-1}{2}}\right]$
9. Right local standard deviation: $\text{std}\left[y_t, \dots, y_{t+\frac{l-1}{2}}\right]$
Output: Feature set.
```

standardized features. Then the random-walk normalized Laplacian is computed, and its first K eigenvectors are collected in a matrix $U \in \mathbb{R}^{T\times K}$. Treating each row of U as a point in $\mathbb{R}^K$, they are grouped into K clusters using the K-means algorithm. Finally, the transition probabilities are computed, by counting the number of transitions between clusters, along with the distributional parameters for each cluster.

Normalizing the rows of U to have unit Euclidean norm before clustering, as proposed by Ng et al. (2002) and advocated by Weiss (1999), does not make a difference. Rather, it is important to standardize the features prior to forming the affinity matrix.

Spectral clustering estimates the parameters and the hidden states simultaneously. Each observation is assigned to a specific state, which is different from the traditional estimation methods that employ probabilistic assignment. Jin, Kang, and Ding (2006) proposed a probabilistic approach for optimizing spectral clustering and argued that soft, as opposed to hard, cluster membership meant that it was less likely to get trapped in a local minimum. However, that is not the impression the comparison in Section 4 with maximum likelihood estimation leaves.

2.1. Feature selection

The performance of spectral clustering is determined by the quality of the eigenvectors of the related graph Laplacian. Generally, the Laplacian is constructed using the full features, which will degrade the quality of the related eigenvectors when there are a large number of noisy or irrelevant features in datasets (Jiang & Ren, 2011). Without feature selection the performance of spectral clustering can degrade dramatically in the presence of irrelevant features (Bach & Jordan, 2004).

Algorithm 2 outlines the features proposed by Zheng et al. (2019) for learning HMMs. In addition to the observations and the left and right absolute changes, they include centered, left, and right local means and standard deviations. Based on a simulation experiment, Zheng et al. (2019) decided to use this feature set with two window lengths $l = 5$ and $l = 13$, i.e., 15 features in total.

The feature set in Algorithm 2 is simple, yet goes beyond the first two moments by including absolute differences. We will use this set in order to make our results directly comparable to those of Zheng et al. (2019). Additionally, we will use the same features as input to the jump estimator proposed in the next section to ensure that spectral clustering and jump estimation have the same starting point. It is left for future research to consider feature selection in depth.

2.2. Affinity matrix

The task of building a good affinity matrix has been shown to be largely responsible for the performance of spectral clustering algorithms (Bach & Jordan, 2004). We follow the most common approach by using the Gaussian kernel function defined by $A_{ij} = \exp(-\gamma \|x_i - x_j\|_2^2)$, where x_i and x_j are two feature vectors. It is simple to calculate and results in a positive definite affinity matrix. The scaling parameter $\gamma = 1/(2\sigma^2)$ controls how rapidly the affinity falls off with the distance between the features. It is a measure of when two feature vectors are considered similar and should be selected for the specific application, for example using cross-validation.

Successful recovery of the hidden partition requires the constructed affinity matrix to have a clear block structure. Fischer and Poland (2005) proposed different ways to amplify the block matrix structure for spectral clustering. Zhang, Li, and Yu (2011) argued that it is desirable that with the same Euclidean distance, two points in the same cluster should have higher similarity than two points in different clusters, which is not the case for the standard Gaussian kernel. This led them to propose a local density adaptive similarity measure that uses the local density between two data points to scale the Gaussian kernel function. Some of the same smoothing effect is achieved through the use of rolling means and standard deviations in the feature calculation in Algorithm 2 as an alternative to considering the observations and their squared values.

2.3. Dimensionality reduction

After the affinity matrix is constructed, its random-walk normalized Laplacian $L_{\text{rw}} = I - D^{-1}A$ is computed (Meila & Shi, 2001). The degree matrix $D = \text{diag}(A1)$, where 1 is a column vector of ones, is a diagonal matrix whose ith diagonal element is the sum of the elements in the ith row of A.

Because the random-walk normalized Laplacian is positive semi-definite, its eigenvectors are real and nonnegative. In most of the literature on spectral clustering it is taken for granted that the eigenvalue problem is easy to solve (Bach & Jordan, 2004). In practice, it is necessary to use an algorithm that exploits the structure of a real symmetric matrix to avoid complex eigenvalues and -vectors as a results of numerical issues. Similarly, the first eigenvalue might be negative as a result of errors introduced by truncation and rounding errors. Several studies have demonstrated that by regularizing the Laplacian the ability of its eigenvectors to discriminate between the clusters can be improved

(Amini, Chen, Bickel, & Levina, 2013; Chaudhuri, Chung, & Tsiatas, 2012; Joseph & Yu, 2016; Qin & Rohe, 2013).

It is a significant advantage of spectral clustering that the dimension of the clustering task only depends on the number of observations and the number of eigenvectors, and is independent of the number of features. Although the number of required eigenvectors is generally equal to the number of clusters, Jin et al. (2006) showed that it can be different when the clusters are not well separated. Zelnik-Manor and Perona (2005) suggested exploiting the structure of the eigenvectors to infer automatically the number of groups. Another possibility is to analyze the eigenvalues. We assume that the number of states K is known and select the same number of eigenvectors.

3. Penalizing jumps to increase persistence

Bemporad et al. (2018) proposed to fit jump models by minimizing the objective

$$\sum_{t=1}^{T-1}\left[\ell(y_t,\theta_{s_t})+\lambda\mathbb{I}_{s_t\neq s_{t+1}}\right]+\ell(y_T,\theta_{s_T}), \tag{1}$$

where $\mathbb{I}$ is the indicator function, over the model parameters $\theta = \{\theta_1,\ldots,\theta_K\}$ and the state sequence $s=\{s_1,\ldots,s_T\}$. This can be viewed as a tradeoff between fitting the data and prior assumptions about the state sequence.

The parameter $\lambda \geq 0$ is the penalty for jumps. For λ large enough, the jump model becomes a single model that covers the whole dataset. When $\lambda = 0$ it reduces to automatically splitting the dataset in K clusters and fitting one model per cluster, which is a generalization of the K-means algorithm of Lloyd (1982). We will consider the loss function $\ell\left(y_t,\theta_{s_t}\right)=\|y_t-\theta_{s_t}\|_2^2$, which is equivalent to K-means clustering.

3.1. Model fitting

Algorithm 3 outlines the proposed jump model approach to estimating HMMs. The inputs are the time series $y=\{y_1,\ldots,y_T\}$, the number of latent states K, and an initial state sequence $s^0 = \left\{s_1^0,\ldots,s_T^0\right\}$. First, a set of standardized features is constructed from the time series. Then, it iterates between finding the parameters that minimize the loss function for a given state sequence—i.e., fitting the model—and finding the state sequence that minimizes the objective function 1 given the parameters—i.e., fitting the state sequence. This process is repeated until the state sequence does not change. In practice, we finish after a maximum of ten iterations or once the value of the objective changes by less than 10^{-6} from one iteration to the next, which usually takes less than five iterations.

Algorithm 3 Jump estimation of HMM.

```
Input: Time series $y=\{y_1,\ldots,y_T\}$, number of latent states $K$, and initial state sequence $s^0=\left\{s_1^0,\ldots,s_T^0\right\}$.
1. Construct a set of standardized features $z$ from the time series $y$.
2. Iterate for $i = 1, \ldots$
   (a) $\theta^i = \operatorname{argmin}_\theta \sum_{t=1}^T \ell\left(z_t,\theta_{s_t^{i-1}}\right)$ (model fitting).
   (b) $s^i = \operatorname{argmin}_s\left\{\sum_{t=1}^{T-1}\left[\ell\left(z_t,\theta_{s_t}^i\right)+\lambda\mathbb{I}_{s_t\neq s_{t+1}}\right]+\ell\left(z_T,\theta_{s_T}^i\right)\right\}$ (state-sequence fitting).
3. Until $s^i = s^{i-1}$.
4. Compute the transition probabilities and distributional parameters for each state.
Output: HMM parameters and prediction of latent states.
```

Based on the final estimate of the state sequence s^i, transition probabilities are computed, by counting the number of transitions between states, along with the distributional parameters for each state. To improve the quality of the solution, we run Algorithm 3 from ten different initial state sequences generated randomly using K-means++ (Arthur & Vassilvitskii, 2007) and keep the model that achieves the lowest objective value.

In Algorithm 3, item 2a, the parameters $\theta^i=\left\{\theta_1^i,\ldots,\theta_K^i\right\}$ are estimated by computing analytically the minimum of the loss function for each of the K states. In item 2b, the most likely sequence of states is found using dynamic programming (Bellman, 1957). Define

$$V(T,s)=\ell(z_T,\theta_s), \tag{2a}$$

$$V(t,i)=\ell(z_t,\theta_i)+\min_j\left[V(t+1,j)+\lambda\mathbb{I}_{i\neq j}\right], \quad t=T-1,\ldots,1. \tag{2b}$$

The most likely sequence of states is then given by

$$s_1=\operatorname*{argmin}_j V(1,j), \tag{3a}$$

$$s_t=\operatorname*{argmin}_j\left[V(t,j)+\lambda\mathbb{I}_{s_{t-1}\neq j}\right], \quad t=2,\ldots,T. \tag{3b}$$

If the time order of operations is reversed, it becomes the Viterbi algorithm (Viterbi, 1967).

Finding the most likely sequence of states requires $O(TK^2)$ operations, which is the same as one forward and backward pass with the EM algorithm. With the same computational complexity per iteration, it is a significant advantage of the jump model that it takes much fewer iterations to converge. In our simulation study, five iterations are typically enough to fit a jump model, whereas the EM algorithm needs between 50 and 100 iterations to converge.

3.2. Selecting the penalty

The jump penalty λ is selected based on the specific application, for example using cross-validation. The penalty aims to regularize the problem (1) by increasing the difference between states, similar to the purpose of amplifying the block structure of the affinity matrix as described in Section 2.2.

The jump penalty summarizes prior knowledge or assumptions about the transition rate. Note that a Bayesian approach would be significantly more demanding computationally. A strong prior is particularly important when a limited number of observations is available. A challenge when interested in fitting highly persistent regime-switching models is that it requires a lot of observations to ensure that all latent states are observed. Hence, a prior or regularization penalty should be imposed.

In many applications of the HMM, asset allocation included, it is a definite advantage of the jump estimator that it is possible to control the transition rate by adjusting the penalty parameter. The penalty parameter can be viewed as a hyperparameter that can be adjusted to achieve the desired behavior. In the case of a trading strategy, tuning the jump penalty is an alternative or supplement to controlling turnover by imposing transaction costs (Boyd et al., 2017; García-Galicia, Carsteanu, & Clempner, 2019; Nystrup et al., 2019). The penalty that yields the most profitable strategy is not necessarily the same value that yields the most accurate estimate of the hidden state sequence. Whereas it would be extremely challenging to select the parameters in an HMM in order to maximize the return of a trading strategy, this is easily done for the jump penalty using cross-validation.

3.3. Variations

It is straightforward to add a regularization term on the parameters θ to (1), but it comes at the cost of not being able to compute analytically the optimal parameters for a given state sequence. Another variation would be to implement a fading memory by adding a decay term to the objective values. Although it is outside the scope of this article, jump model fitting can be extended to streaming applications, as proposed by Bemporad et al. (2018).

When penalizing jumps it is no longer necessary to construct an affinity matrix and compute its Laplacian. Better results can be achieved by fitting jump models directly to the standardized temporal features in Algorithm 2. It is still possible, though, to consider the first few eigenvectors of the Laplacian if dimensionality reduction is needed. Jump fitting could even be applied as a post-processing step after spectral clustering.

Bemporad et al. (2018) showed how the jump framework can be used for fitting HMMs by replacing the least-squares criterion by the likelihood function and selecting the jump penalty based on transition probabilities. After all, simple features constructed from the data cannot add information compared to using the correctly specified likelihood function. However, when using the likelihood objective the choice of the jump penalty becomes critical and can no longer be viewed as a regularization parameter. For λ large enough, the model never changes state and the accuracy converges to the reciprocal of the number of states, similar to when using (1). When λ is too small, it becomes a mixture model with no temporal information and the accuracy, once again, converges to the reciprocal of the number of states. This is crucially different from our proposal of using (1) based on temporal features, where too small of a value of λ means that the outcome essentially converges to the result of spectral clustering.

Learning HMMs by minimizing the least-squares criterion (1) is more robust than likelihood maximization, because it does not rely on distributional assumptions. Moreover, the method is expected to be more robust to initialization and an increasing number of states, as was the case with spectral clustering (Zheng et al., 2019). The potential for adding exogenous variables that could improve latent state prediction is perhaps the most exciting prospect. Similar to when using spectral clustering, it is straightforward to include additional features, because these do not have to be multivariate Gaussian.

4. Simulation study

We compare the accuracy of HMMs estimated using maximum likelihood, spectral, and jump estimation in a simulation study. The advantage of a simulation study is that the true state sequence is known, which makes it possible to evaluate the ability of each model to correctly identify the underlying hidden states.

Accuracy is defined as the maximum classification accuracy among all possible alignments of an inferred state sequence. It is necessary to check whether a different permutation of the hidden states yields higher accuracy due to the risk of label switching. It is the ratio tp/(tp + fn), where tp is the number of true positives and fn the number of false negatives. We use the balanced accuracy (BAC), which is the average of accuracy per observed state, to avoid inflated performance estimates on imbalanced datasets (Brodersen, Ong, Stephan, & Buhmann, 2010). If a classifier performs equally well on either state, BAC reduces to conventional accuracy. In contrast, if accuracy is above chance only because a classifier takes advantage of an imbalanced dataset, then balanced accuracy, as appropriate, drops to the reciprocal of the number of states.

4.1. Two-state HMM

Following Zheng et al. (2019), we simulate data from a two-state Gaussian HMM

$$y_t | s_t \sim N\left(\mu_{s_t}, \sigma_{s_t}^2\right), \tag{4}$$

where s_t is a first-order Markov chain, with parameters

$$\begin{aligned} \mu_1 &= 0.0123, \quad \mu_2 = -0.0157, \\ \sigma_1 &= 0.0347, \quad \sigma_2 = 0.0778, \\ \Gamma &= \begin{pmatrix} 0.9629 & 0.0371 \\ 0.2101 & 0.7899 \end{pmatrix}. \end{aligned} \tag{5}$$

This is a model Hardy (2001) estimated from monthly returns on a stock index. As shown by Rydén et al. (1998), this model captures the stylized behavior of many financial time series including volatility clustering and leptokurtosis. The significant overlap between the conditional distributions makes it challenging to infer the unobserved state sequence.

Each series is simulated by first generating a sequence of states of desired length according to the transition probability matrix Γ starting from its stationary distribution. Secondly, observations are sampled from conditional Gaussian distributions with mean and volatility parameters μ and σ given by the simulated states.

To simulate data from models with different levels of persistence in the state process, we assume the following relations between the model parameters at two time scales t_1 and t_2:

$$\begin{aligned} \Gamma(t_1)^{1/t_1} &= \Gamma(t_2)^{1/t_2}, \\ \mu_s(t_1)/t_1 &= \mu_s(t_2)/t_2, \\ \sigma_s(t_1)/\sqrt{t_1} &= \sigma_s(t_2)/\sqrt{t_2}, \end{aligned}$$

where $\Gamma(t)$, $\mu(t)$, and $\sigma(t)$ are the model parameters associated with time scale t. These relations are implied by assuming the instantaneous log return follows a Lévy process, i.e., a general continuous-time random walk with independent, stationary increments. The signal-to-noise ratio decreases with the time scale while persistence increases. In addition to the monthly base frequency $t = 20$ with parameters given in (5), we consider the weekly $t = 5$ and daily scale $t = 1$.

4.1.1. Selecting γ and λ

Fig. 1 shows the BAC of spectral clustering as a function of the scaling parameter γ used in the Gaussian kernel function for different simulation lengths using the daily parameter values. The

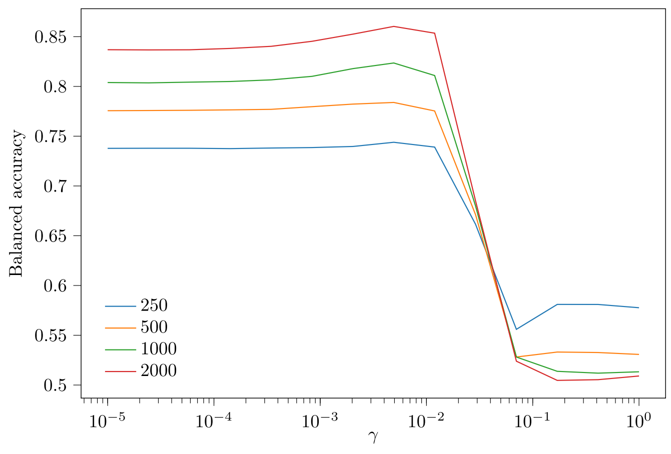

Fig. 1. Balanced accuracy of spectral clustering as a function of the scaling parameter used in the Gaussian kernel function for different simulation lengths using the daily parameter values.

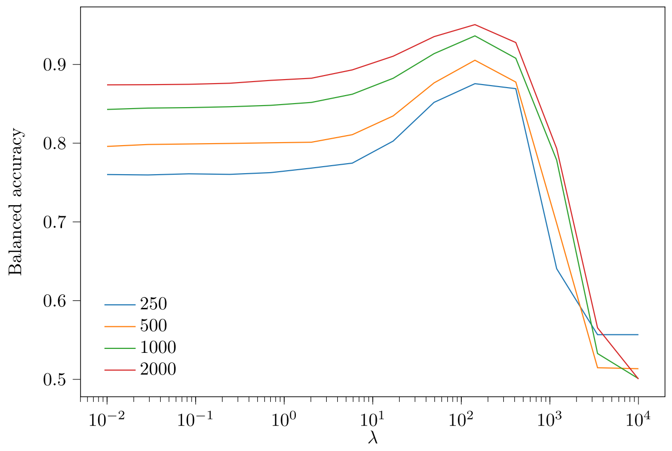

Fig. 2. Balanced accuracy of jump estimation as a function of the penalty parameter for different simulation lengths using the daily parameter values.

longer the simulated series, the higher the BAC. Across simulation lengths, any value $\gamma < 10^{-2}$ works. The corresponding figures for the weekly and monthly parameters look similar, with the only difference being the level of BAC. We use $\gamma = 10^{-3}$ throughout the simulation study, because the difference compared to using the optimal value for each time scale and simulation length is negligible. Using a common value across all simulation settings increases robustness of the results.

Fig. 2 shows the BAC of jump estimation as a function of the penalty parameter λ for different simulation lengths using the daily parameter values. The main difference between simulation lengths is the level of BAC, with longer series leading to higher BAC. When the value of the penalty parameter is too large, the BAC approaches 1/2. When the penalty parameter is too small, the BAC converges to that of the unpenalized case, which corresponds to applying K-means to the feature matrix. Across simulation lengths at the daily scale, the optimal value of the penalty parameter is around $\lambda = 10^2$. Without showing the corresponding figures for the weekly and monthly scales, we note that the optimal values are around $\lambda = 50$ and $\lambda = 1$, respectively. The less persistent the state process, the lower the optimal value of the penalty parameter and the smaller the improvement in BAC that is achieved by penalizing jumps.

4.1.2. Daily scale

Table 1 shows the mean and standard deviation (in parentheses) of the estimated parameters based on 1000 series of different lengths simulated using the daily equivalent of the model 5. Bold entries identify the best estimate between maximum likelihood, spectral, and jump. The accuracy using the true parameters is based on the Viterbi path.

The reported MLEs are obtained using the regularized EM algorithm based on Huang, Acero, and Hon (2001, Chapter 9.6.1) that is implemented in the `hmmlearn` package in Python with the variance prior set to 10^{-4}. The MLE is reported for two different initializations: MLE$_K$ is initialized using K-means with ten different centroid seeds generated using K-means++ (Arthur & Vassilvitskii, 2007), similar to how spectral clustering and jump estimation are initialized. MLE is initialized with a probability 1 of being in the first state, conditional means $\mu_1 = \mu_2 = 0$, standard deviations $\sigma_1 = \sigma_2 = 0.01$, and self-transition probabilities of 0.9.

We use the EM algorithm rather than direct numerical maximization of the likelihood function, because this is known to be more robust to initial values. Even so, at the shortest simulations lengths, the initialization makes a big difference in terms of the estimated transition probabilities and accuracy with MLE being much better than MLE$_K$. At the shortest simulation lengths, spectral clustering with $\gamma = 10^{-3}$ leads to much better estimates of the transition probabilities compared to MLE$_K$, but lower BACs. Jump estimation with $\lambda = 10^2$ yields more accurate parameter estimates compared to the other methods and a BAC inline with the best MLE.

It is arguably more fair to compare spectral clustering and jump estimation to MLE$_K$, which is initialized the same way and does not have the advantage of better initial values. The much higher standard deviation of the MLE$_K$ parameter estimates is evidence of maximum likelihood estimation being less robust than spectral clustering and jump estimation.

The estimates of the daily parameters are plotted in Fig. 3. It is clear from the figure that the MLE and the jump estimate are the fastest to converge to the true values as the simulation length increases. The jump estimate particularly stands out when looking at the transition probabilities in the bottom row of the figure by

Table 1
Mean and standard deviation (in parentheses) of estimated parameters based on 1000 series of different lengths simulated using the daily parameter values. Bold entries identify the best estimate between MLE, spectral, and jump.

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC
250									
True	0.0006	−0.0008	0.0078	0.0174	0.0021	0.0120	0.9932 (0.0404)	0.8757 (0.2469)	0.9344 (0.0587)
MLE$_K$	0.0008 (0.0022)	−0.0001 (0.0043)	0.0075 (0.0017)	0.0119 (0.0048)	0.2889 (0.2443)	0.3694 (0.2864)	0.7689 (0.2105)	0.8101 (0.2505)	0.7895 (0.0206)
MLE	**0.0007** (0.0012)	0.0000 (0.0047)	**0.0077** (0.0012)	0.0123 (0.0048)	0.0453 (0.0573)	0.1180 (0.1645)	**0.9250** (0.1384)	0.8686 (0.2417)	**0.8968** (0.0282)
Spectral	0.0010 (0.0024)	−**0.0006** (0.0044)	0.0073 (0.0022)	0.0121 (0.0048)	0.0933 (0.0520)	0.1605 (0.0600)	0.7384 (0.1845)	0.7405 (0.1984)	0.7394 (0.0011)
Jump	**0.0007** (0.0010)	0.0000 (0.0036)	**0.0079** (0.0016)	**0.0126** (0.0051)	**0.0056** (0.0060)	**0.0278** (0.0325)	0.8766 (0.1491)	**0.8772** (0.2017)	0.8769 (0.0003)
500									
True	0.0006	−0.0008	0.0078	0.0174	0.0021	0.0120	0.9968 (0.0089)	0.8662 (0.2514)	0.9315 (0.0653)
MLE$_K$	0.0007 (0.0012)	−0.0002 (0.0022)	0.0075 (0.0009)	0.0135 (0.0047)	0.2025 (0.2334)	0.2513 (0.2619)	0.8421 (0.1992)	0.8592 (0.2076)	0.8507 (0.0086)
MLE	**0.0006** (0.0006)	−0.0004 (0.0032)	0.0077 (0.0004)	0.0138 (0.0047)	0.0303 (0.0427)	0.0762 (0.1077)	**0.9591** (0.0951)	0.8770 (0.2271)	**0.9181** (0.0410)
Spectral	0.0008 (0.0018)	−**0.0008** (0.0039)	0.0072 (0.0013)	0.0136 (0.0049)	0.0770 (0.0527)	0.1518 (0.0594)	0.7874 (0.1856)	0.7734 (0.1883)	0.7804 (0.0070)
Jump	**0.0006** (0.0006)	−0.0004 (0.0029)	**0.0078** (0.0004)	**0.0143** (0.0048)	**0.0036** (0.0029)	**0.0237** (0.0249)	0.9243 (0.1214)	**0.8816** (0.1903)	0.9029 (0.0214)
1000									
True	0.0006	−0.0008	0.0078	0.0174	0.0021	0.0120	0.9962 (0.0076)	0.8993 (0.1915)	0.9478 (0.0485)
MLE$_K$	**0.0006** (0.0007)	−0.0005 (0.0017)	0.0076 (0.0004)	0.0154 (0.0039)	0.1019 (0.1881)	0.1371 (0.2165)	0.9274 (0.1475)	0.8805 (0.1817)	0.9039 (0.0235)
MLE	**0.0006** (0.0004)	−0.0005 (0.0021)	0.0077 (0.0003)	0.0159 (0.0036)	0.0157 (0.0315)	0.0395 (0.0583)	**0.9820** (0.0627)	0.8921 (0.2052)	**0.9370** (0.0450)
Spectral	0.0009 (0.0014)	−0.0014 (0.0032)	0.0073 (0.0009)	0.0152 (0.0043)	0.0565 (0.0472)	0.1434 (0.0586)	0.8467 (0.1654)	0.7774 (0.1705)	0.8120 (0.0346)
Jump	**0.0006** (0.0003)	−**0.0008** (0.0025)	**0.0078** (0.0003)	**0.0164** (0.0035)	**0.0025** (0.0015)	**0.0192** (0.0176)	0.9706 (0.0772)	**0.8928** (0.1623)	0.9317 (0.0389)
2000									
True	0.0006	−0.0008	0.0078	0.0174	0.0021	0.0120	0.9965 (0.0055)	0.9264 (0.1198)	0.9614 (0.0350)
MLE$_K$	**0.0006** (0.0003)	−**0.0007** (0.0011)	0.0077 (0.0003)	0.0166 (0.0025)	0.0348 (0.1077)	0.0664 (0.1480)	0.9791 (0.0759)	**0.9134** (0.1273)	0.9463 (0.0329)
MLE	**0.0006** (0.0002)	−**0.0007** (0.0013)	**0.0078** (0.0003)	0.0169 (0.0022)	0.0071 (0.0209)	0.0229 (0.0279)	**0.9946** (0.0199)	0.9036 (0.1876)	**0.9491** (0.0455)
Spectral	0.0009 (0.0007)	−0.0017 (0.0023)	0.0074 (0.0006)	0.0164 (0.0032)	0.0387 (0.0301)	0.1421 (0.0489)	0.9021 (0.1187)	0.7918 (0.1281)	0.8470 (0.0552)
Jump	**0.0006** (0.0002)	−0.0010 (0.0017)	**0.0078** (0.0002)	**0.0174** (0.0018)	**0.0021** (0.0010)	**0.0165** (0.0123)	0.9928 (0.0246)	0.9020 (0.1161)	0.9474 (0.0454)

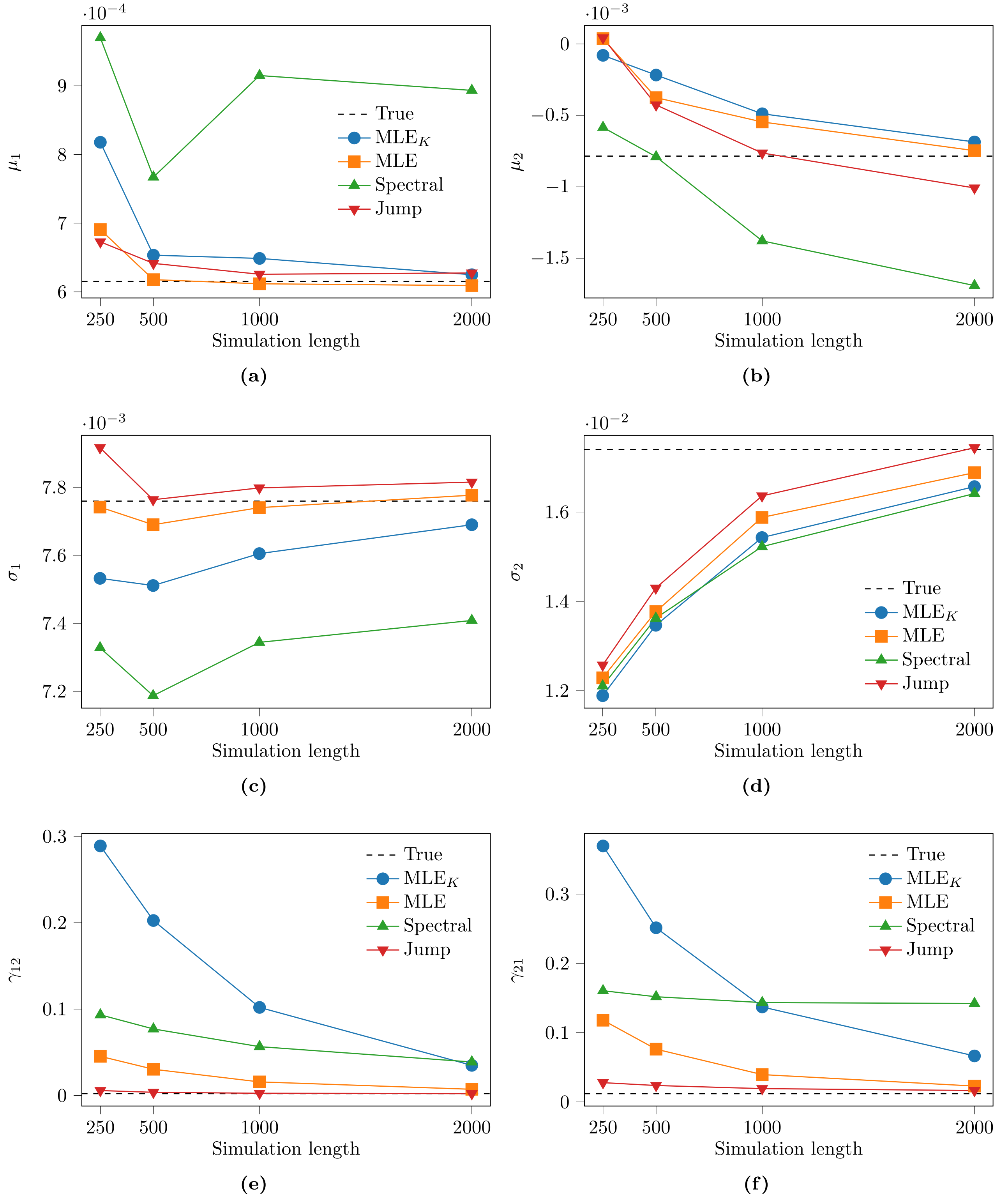

Fig. 3. Estimates of daily parameters based on 1000 simulations as a function of the simulation length using different estimation approaches.

being very close to the true values even for simulated series of only 250 observations. Even with 2000 observations per simulation, MLE$_K$ and spectral clustering are far from estimating the true persistence of the state process.

When simulating a series of 250 observations using the daily parameter values, the probability that it only includes observations from one state is 51%. When simulating a series of length 500, the probability is 30%, for 1000 observations it is 10%, and for 2000 observations it is 1%.[1] In their simulation study, Zheng et al. (2019) filtered the simulated series to exclude those that only included observations from one state. If one state is unobserved in a sample, the maximum likelihood problem is non-

[1] The probabilities are calculated analytically using the fact that sojourn times are geometrically distributed.

Table 2
Mean and standard deviation (in parentheses) of estimated parameters based on 1000 series of different lengths simulated using the weekly parameter values. Bold entries identify the best estimate between MLE, spectral, and jump.

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC
250									
True	0.0031	−0.0039	0.0174	0.0389	0.0103	0.0582	0.9858 (0.0267)	0.6890 (0.2999)	0.8374 (0.1484)
MLE_K	0.0033 (0.0033)	−0.0030 (0.0104)	0.0166 (0.0018)	0.0334 (0.0106)	0.1272 (0.1975)	0.2472 (0.2682)	0.8945 (0.1596)	0.6997 (0.2542)	0.7971 (0.0974)
MLE	**0.0031** (0.0017)	−0.0031 (0.0113)	**0.0171** (0.0013)	0.0342 (0.0100)	0.0296 (0.0409)	0.1234 (0.1478)	**0.9517** (0.0915)	0.7064 (0.2883)	0.8291 (0.1226)
Spectral	0.0038 (0.0032)	−**0.0043** (0.0104)	0.0165 (0.0021)	0.0339 (0.0104)	0.0547 (0.0443)	0.1484 (0.0628)	0.8483 (0.1531)	**0.7649** (0.2114)	0.8066 (0.0417)
Jump	**0.0031** (0.0018)	−0.0034 (0.0101)	**0.0177** (0.0016)	**0.0358** (0.0101)	**0.0090** (0.0056)	**0.0509** (0.0317)	0.9291 (0.1113)	0.7455 (0.2502)	**0.8373** (0.0918)
500									
True	0.0031	−0.0039	0.0174	0.0389	0.0103	0.0582	0.9865 (0.0176)	0.7587 (0.1884)	0.8726 (0.1139)
MLE_K	0.0032 (0.0015)	−0.0037 (0.0071)	0.0170 (0.0010)	0.0367 (0.0071)	0.0524 (0.1200)	0.1348 (0.1729)	0.9566 (0.0959)	0.7554 (0.1869)	0.8560 (0.1006)
MLE	**0.0031** (0.0010)	−**0.0040** (0.0079)	**0.0172** (0.0008)	0.0372 (0.0063)	0.0165 (0.0195)	0.0892 (0.0956)	**0.9787** (0.0386)	0.7544 (0.2077)	**0.8665** (0.1122)
Spectral	0.0039 (0.0020)	−0.0062 (0.0078)	0.0166 (0.0015)	0.0358 (0.0080)	0.0433 (0.0318)	0.1516 (0.0515)	0.8890 (0.1181)	0.7576 (0.1608)	0.8233 (0.0657)
Jump	0.0032 (0.0011)	−0.0054 (0.0072)	0.0178 (0.0010)	**0.0385** (0.0061)	**0.0074** (0.0034)	**0.0508** (0.0227)	0.9703 (0.0558)	**0.7584** (0.1754)	0.8643 (0.1060)
1000									
True	0.0031	−0.0039	0.0174	0.0389	0.0103	0.0582	0.9879 (0.0113)	0.7732 (0.1165)	0.8805 (0.1073)
MLE_K	**0.0031** (0.0007)	−0.0036 (0.0038)	0.0172 (0.0006)	0.0380 (0.0040)	0.0196 (0.0441)	0.0884 (0.0968)	0.9845 (0.0217)	**0.7714** (0.1322)	**0.8780** (0.1065)
MLE	**0.0031** (0.0006)	−**0.0037** (0.0038)	**0.0173** (0.0007)	0.0381 (0.0038)	**0.0129** (0.0120)	0.0726 (0.0457)	**0.9861** (0.0147)	0.7650 (0.1576)	0.8755 (0.1105)
Spectral	0.0039 (0.0012)	−0.0069 (0.0056)	0.0168 (0.0010)	0.0371 (0.0055)	0.0357 (0.0199)	0.1573 (0.0412)	0.9194 (0.0825)	0.7515 (0.1065)	0.8355 (0.0840)
Jump	0.0032 (0.0007)	−0.0057 (0.0053)	0.0178 (0.0006)	**0.0394** (0.0035)	0.0067 (0.0025)	**0.0486** (0.0171)	0.9818 (0.0194)	0.7561 (0.1203)	0.8689 (0.1129)
2000									
True	0.0031	−0.0039	0.0174	0.0389	0.0103	0.0582	0.9882 (0.0079)	0.7912 (0.0757)	0.8897 (0.0985)
MLE_K	**0.0031** (0.0004)	−0.0041 (0.0024)	0.0173 (0.0003)	**0.0387** (0.0021)	**0.0111** (0.0053)	0.0648 (0.0289)	0.9877 (0.0090)	**0.7896** (0.0829)	**0.8887** (0.0991)
MLE	**0.0031** (0.0005)	−**0.0040** (0.0026)	**0.0174** (0.0007)	0.0384 (0.0031)	0.0125 (0.0123)	**0.0644** (0.0213)	**0.9879** (0.0090)	0.7763 (0.1327)	0.8821 (0.1058)
Spectral	0.0039 (0.0006)	−0.0080 (0.0036)	0.0169 (0.0007)	0.0379 (0.0038)	0.0323 (0.0119)	0.1635 (0.0314)	0.9382 (0.0460)	0.7427 (0.0761)	0.8404 (0.0977)
Jump	0.0032 (0.0005)	−0.0065 (0.0032)	0.0179 (0.0005)	0.0398 (0.0023)	0.0067 (0.0017)	0.0478 (0.0118)	0.9846 (0.0080)	0.7565 (0.0870)	0.8706 (0.1140)

regular. However, excluding the most persistent samples inevitably introduces a bias in the transition probabilities.

In considering regime-switching models with high persistence, it is an issue that it requires a lot of observations to ensure that all latent states are featured. In practical applications, for example when using a rolling window for estimation, it might not be the case that all states are observed. It is important how an estimator performs in that situation and, therefore, those series should not be excluded. It reflects negatively on the average BAC when fitting a two-state model based on data from one state only, but it is important that the estimator does not break down. To that end, the regularization imbedded in the jump estimator and the variance prior in the chosen EM algorithm are pivotal. For a sufficiently large penalty on jumps, the estimator has no problem identifying when all observations originate from the same conditional distribution.

Comparing our results when simulating series of 500 observations to those of Zheng et al. (2019), the MLE has a higher accuracy while spectral clustering has a lower accuracy. The higher accuracy of MLE in our study can be explained by the choice of the more robust EM algorithm instead of direct numerical maximization of the likelihood function. This may also explain their choice of filtering the simulated series to exclude those that only include observations from one state. The results in Table 1 show that there is no need for filtering the simulated data, since the BACs of MLE and jump are close to those of the Viterbi path using the true parameters.

4.1.3. Weekly scale

Table 2 shows the mean and standard deviation (in parentheses) of the estimated parameters based on 1000 series of different lengths simulated using the weekly equivalent of the model (5). Given the lower persistence of the state process at the weekly scale, it is not longer likely that a sequence only includes observations from one state, even if it only consists of 250 observations. After all, 250 weeks correspond to a period of five years. Results are shown for $\lambda = 50$. Naturally, the optimal value of the jump penalty is lower, because the state process is less persistent.

The lower persistence leads to lower BACs, meaning the hidden states are harder to come by. This is also true for the Viterbi path based on the true parameter values. In terms of BAC, MLE_K has the highest value at the longest simulation lengths, but only by a narrow margin over the other estimators. Compared to at the daily scale, the different estimation methods are better at estimating the transition probabilities, with the jump estimates still being by far the best. The BAC of MLE and jump is close to what can be achieved using the true parameter values.

4.1.4. Monthly scale

Table 3 shows the mean and standard deviation (in parentheses) of the estimated parameters based on 1000 series of different lengths simulated using the monthly parameters (5). Results are shown for $\lambda = 1$.

The relatively low persistence at the monthly scale leads to lower BACs. Spectral clustering and jump estimation are much better than MLE both in terms of parameter estimates and BAC. It is remarkable that both estimators yield a higher BAC than the Viterbi path based on the true parameter values. Even though the motivation for penalizing jumps is to improve estimates of transition probabilities for persistent state processes, it also leads to much better estimates of the less persistent process at the monthly scale. It is evident that MLE and the Viterbi algorithm have a hard time predicting the second, less persistent state with accuracy for this state not much above 1/2.

4.2. Three-state HMM

It is well known that HMMs become considerably harder to estimate as the number of hidden states increases. To compare the different estimation methods when estimating a model with three hidden states, we follow Zheng et al. (2019) by augmenting the two-state model (5) with an intermediate third state:

$$\mu_1 = 0.0123, \quad \mu_2 = 0.0000, \quad \mu_3 = -0.0157,$$
$$\sigma_1 = 0.0347, \quad \sigma_2 = 0.0500, \quad \sigma_3 = 0.0778,$$
$$\Gamma = \begin{pmatrix} 0.9629 & 0.0185 & 0.0186 \\ 0.0618 & 0.8764 & 0.0618 \\ 0.1051 & 0.1050 & 0.7899 \end{pmatrix}. \tag{6}$$

Models with more than two states are typically preferred when considering long series of returns (Dias et al., 2015; Nystrup et al., 2015b).

Table 3
Mean and standard deviation (in parentheses) of estimated parameters based on 1000 series of different lengths simulated using the monthly parameter values. Bold entries identify the best estimate between MLE, spectral, and jump.

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC
250									
True	0.0123	−0.0157	0.0347	0.0778	0.0371	0.2101	0.9845 (0.0194)	0.4914 (0.1882)	0.7380 (0.2465)
MLE$_K$	0.0127 (0.0045)	−0.0194 (0.0303)	0.0338 (0.0032)	0.0708 (0.0176)	0.0942 (0.1257)	0.3389 (0.2338)	0.9382 (0.1037)	0.5234 (0.2089)	0.7308 (0.2074)
MLE	**0.0124** (0.0032)	−0.0187 (0.0313)	**0.0344** (0.0033)	**0.0711** (0.0178)	0.0513 (0.0460)	0.2651 (0.1931)	**0.9544** (0.0775)	0.5157 (0.2320)	0.7351 (0.2193)
Spectral	0.0146 (0.0049)	−**0.0158** (0.0184)	0.0332 (0.0034)	0.0676 (0.0152)	0.0539 (0.0344)	**0.1735** (0.0556)	0.8489 (0.1150)	0.7029 (0.1737)	0.7759 (0.0730)
Jump	0.0143 (0.0050)	−0.0165 (0.0176)	0.0333 (0.0031)	0.0688 (0.0154)	**0.0425** (0.0233)	0.1494 (0.0471)	0.8695 (0.1072)	**0.7163** (0.1568)	**0.7929** (0.0766)
500									
True	0.0123	−0.0157	0.0347	0.0778	0.0371	0.2101	0.9833 (0.0136)	0.5230 (0.1230)	0.7532 (0.2301)
MLE$_K$	0.0125 (0.0021)	−0.0171 (0.0148)	0.0343 (0.0018)	**0.0758** (0.0101)	0.0495 (0.0487)	0.2541 (0.1323)	0.9729 (0.0389)	0.5266 (0.1535)	0.7498 (0.2231)
MLE	**0.0124** (0.0020)	−**0.0167** (0.0147)	**0.0344** (0.0019)	0.0756 (0.0102)	0.0423 (0.0222)	**0.2332** (0.1091)	**0.9732** (0.0358)	0.5283 (0.1587)	0.7507 (0.2225)
Spectral	0.0145 (0.0031)	−0.0183 (0.0129)	0.0335 (0.0024)	0.0694 (0.0107)	0.0480 (0.0225)	0.1802 (0.0431)	0.8758 (0.0811)	0.6921 (0.1240)	0.7839 (0.0919)
Jump	0.0146 (0.0029)	−0.0193 (0.0114)	0.0335 (0.0020)	0.0703 (0.0101)	**0.0390** (0.0151)	0.1583 (0.0345)	0.8946 (0.0681)	**0.7197** (0.1095)	**0.8071** (0.0875)
1000									
True	0.0123	−0.0157	0.0347	0.0778	0.0371	0.2101	0.9838 (0.0095)	0.5302 (0.0873)	0.7570 (0.2268)
MLE$_K$	0.0124 (0.0014)	−**0.0160** (0.0088)	0.0345 (0.0012)	**0.0769** (0.0065)	0.0409 (0.0214)	0.2262 (0.0753)	**0.9794** (0.0218)	0.5337 (0.1131)	0.7566 (0.2229)
MLE	**0.0123** (0.0015)	−0.0153 (0.0094)	**0.0347** (0.0019)	0.0762 (0.0081)	0.0408 (0.0160)	**0.2171** (0.0691)	**0.9794** (0.0177)	0.5244 (0.1373)	0.7519 (0.2275)
Spectral	0.0146 (0.0019)	−0.0193 (0.0080)	0.0336 (0.0016)	0.0699 (0.0076)	0.0463 (0.0149)	0.1883 (0.0318)	0.8898 (0.0534)	0.6911 (0.0870)	0.7904 (0.0994)
Jump	0.0145 (0.0017)	−0.0206 (0.0073)	0.0336 (0.0014)	0.0710 (0.0071)	**0.0374** (0.0101)	0.1645 (0.0261)	0.9082 (0.0419)	**0.7147** (0.0756)	**0.8114** (0.0968)
2000									
True	0.0123	−0.0157	0.0347	0.0778	0.0371	0.2101	0.9842 (0.0065)	0.5291 (0.0648)	0.7567 (0.2275)
MLE$_K$	**0.0124** (0.0009)	−**0.0161** (0.0059)	**0.0346** (0.0008)	**0.0772** (0.0043)	0.0379 (0.0092)	0.2172 (0.0465)	0.9827 (0.0100)	0.5305 (0.0826)	0.7566 (0.2261)
MLE	**0.0122** (0.0012)	−0.0151 (0.0075)	0.0350 (0.0022)	0.0759 (0.0076)	0.0401 (0.0151)	**0.2105** (0.0499)	**0.9832** (0.0103)	0.5099 (0.1316)	0.7465 (0.2367)
Spectral	0.0146 (0.0012)	−0.0199 (0.0056)	0.0337 (0.0011)	0.0700 (0.0051)	0.0448 (0.0103)	0.1924 (0.0222)	0.8982 (0.0343)	0.6899 (0.0607)	0.7941 (0.1041)
Jump	0.0145 (0.0012)	−0.0213 (0.0051)	0.0337 (0.0010)	0.0710 (0.0049)	**0.0367** (0.0075)	0.1683 (0.0192)	0.9137 (0.0291)	**0.7129** (0.0532)	**0.8133** (0.1004)

Table 4
Mean and standard deviation (in parentheses) of estimated parameters based on 1000 series of different lengths simulated using the daily parameter values. Bold entries identify the best estimate between MLE, spectral, and jump.

	μ_1	μ_2	μ_3	σ_1	σ_2	σ_3	γ_{12}	γ_{13}
500								
True	0.0006	0.0000	−0.0008	0.0078	0.0112	0.0174	0.0009	0.0010
MLE$_K$	0.0003 (0.0030)	0.0003 (0.0035)	0.0001 (0.0035)	0.0094 (0.0038)	0.0093 (0.0030)	0.0104 (0.0039)	0.2256 (0.1843)	0.2115 (0.1795)
MLE	**0.0005** (0.0009)	**0.0001** (0.0027)	−0.0002 (0.0022)	**0.0087** (0.0027)	**0.0114** (0.0037)	0.0126 (0.0044)	0.0219 (0.0302)	0.0243 (0.0233)
Spectral	**0.0007** (0.0038)	−**0.0001** (0.0063)	−0.0001 (0.0067)	0.0068 (0.0026)	0.0096 (0.0033)	0.0123 (0.0048)	0.0938 (0.0483)	0.0712 (0.0465)
Jump	**0.0005** (0.0026)	−**0.0001** (0.0048)	−**0.0004** (0.0043)	0.0088 (0.0031)	0.0115 (0.0041)	**0.0134** (0.0049)	**0.0040** (0.0089)	**0.0050** (0.0114)
1000								
True	0.0006	0.0000	−0.0008	0.0078	0.0112	0.0174	0.0009	0.0010
MLE$_K$	0.0001 (0.0018)	0.0003 (0.0019)	0.0004 (0.0020)	0.0115 (0.0042)	0.0091 (0.0029)	0.0093 (0.0030)	0.1290 (0.1628)	0.1498 (0.1790)
MLE	0.0005 (0.0006)	−0.0001 (0.0016)	−0.0002 (0.0022)	0.0084 (0.0021)	0.0125 (0.0036)	0.0134 (0.0041)	0.0156 (0.0222)	0.0181 (0.0229)
Spectral	**0.0006** (0.0031)	0.0006 (0.0066)	−**0.0008** (0.0068)	0.0065 (0.0017)	0.0098 (0.0030)	0.0131 (0.0047)	0.0947 (0.0484)	0.0574 (0.0423)
Jump	**0.0006** (0.0018)	**0.0000** (0.0049)	−0.0004 (0.0041)	**0.0083** (0.0021)	**0.0124** (0.0036)	**0.0150** (0.0044)	**0.0024** (0.0051)	**0.0028** (0.0060)
2000								
True	0.0006	0.0000	−0.0008	0.0078	0.0112	0.0174	0.0009	0.0010
MLE$_K$	−0.0003 (0.0012)	0.0005 (0.0011)	0.0005 (0.0012)	0.0133 (0.0034)	0.0087 (0.0023)	0.0085 (0.0022)	0.0518 (0.1082)	0.0796 (0.1487)
MLE	**0.0006** (0.0003)	−0.0002 (0.0014)	−0.0003 (0.0011)	**0.0081** (0.0014)	0.0132 (0.0030)	0.0139 (0.0034)	0.0108 (0.0188)	0.0122 (0.0203)
Spectral	**0.0006** (0.0028)	0.0004 (0.0069)	−**0.0009** (0.0069)	0.0065 (0.0008)	**0.0096** (0.0026)	0.0141 (0.0038)	0.1005 (0.0482)	0.0445 (0.0351)
Jump	**0.0006** (0.0010)	−**0.0001** (0.0046)	−0.0010 (0.0039)	0.0082 (0.0014)	0.0131 (0.0030)	**0.0165** (0.0030)	**0.0016** (0.0026)	**0.0016** (0.0031)

	γ_{21}	γ_{23}	γ_{31}	γ_{32}	Accuracy 1	Accuracy 2	Accuracy 3	BAC
500								
True	0.0032	0.0037	0.0058	0.0062	0.9534 (0.1737)	0.7417 (0.3857)	0.7734 (0.3464)	0.8228 (0.0932)
MLE$_K$	0.2594 (0.2129)	0.2784 (0.1870)	0.2478 (0.1990)	0.2844 (0.1830)	0.6250 (0.2832)	0.3879 (0.3587)	0.3415 (0.4017)	0.4515 (0.1242)
MLE	0.0370 (0.0507)	0.0444 (0.0553)	0.0346 (0.0340)	0.0381 (0.0312)	**0.8726** (0.2790)	0.5957 (0.4203)	**0.6886** (0.3996)	**0.7190** (0.1151)
Spectral	0.1318 (0.0493)	0.0999 (0.0547)	0.1232 (0.0519)	0.1176 (0.0475)	0.5912 (0.2264)	0.4583 (0.1926)	0.5634 (0.1963)	0.5377 (0.0572)
Jump	**0.0123** (0.0205)	**0.0158** (0.0257)	**0.0142** (0.0210)	**0.0154** (0.0231)	0.7839 (0.2811)	**0.6035** (0.3631)	0.6773 (0.2635)	0.6883 (0.0741)
1000								
True	0.0032	0.0037	0.0058	0.0062	0.9727 (0.1021)	0.7545 (0.3489)	0.8205 (0.2829)	0.8492 (0.0913)
MLE$_K$	0.1491 (0.1969)	0.3399 (0.1895)	0.1644 (0.1972)	0.3475 (0.1919)	0.7457 (0.3094)	0.2527 (0.3732)	0.1628 (0.3379)	0.3871 (0.2562)
MLE	0.0268 (0.0403)	0.0381 (0.0322)	0.0292 (0.0482)	0.0387 (0.0362)	**0.9007** (0.2446)	0.5177 (0.4253)	0.6385 (0.4185)	0.6856 (0.1599)
Spectral	0.1491 (0.0432)	0.0942 (0.0525)	0.1193 (0.0439)	0.1207 (0.0465)	0.6290 (0.2002)	0.4218 (0.1721)	0.5644 (0.1979)	0.5384 (0.0866)
Jump	**0.0112** (0.0153)	**0.0154** (0.0209)	**0.0126** (0.0157)	**0.0157** (0.0169)	0.8777 (0.2299)	**0.5467** (0.3771)	**0.6612** (0.2478)	**0.6952** (0.1373)
2000								
True	0.0032	0.0037	0.0058	0.0062	0.9834 (0.0355)	0.8101 (0.2732)	0.8453 (0.2234)	0.8796 (0.0748)
MLE$_K$	0.0613 (0.1422)	0.4085 (0.1686)	0.0889 (0.1731)	0.4132 (0.1721)	0.8662 (0.2863)	0.1360 (0.3157)	0.0681 (0.2313)	0.3568 (0.3613)
MLE	0.0172 (0.0266)	0.0356 (0.0229	0.0190 (0.0206)	0.0371 (0.0253)	0.9304 (0.1794)	0.4930 (0.3787)	0.6171 (0.3983)	0.6802 (0.1840)
Spectral	0.1627 (0.0359)	0.0822 (0.0500)	0.1170 (0.0395)	0.1233 (0.0374)	0.6608 (0.1671)	0.3969 (0.1381)	0.5685 (0.1805)	0.5421 (0.1094)
Jump	**0.0095** (0.0112)	**0.0143** (0.0170	**0.0108** (0.0104)	**0.0164** (0.0137)	**0.9484** (0.1628)	**0.5324** (0.3752)	**0.6590** (0.2301)	**0.7132** (0.1741)

The figures corresponding to Fig. 1 and Fig. 2 look similar when simulating from a model with three states, except for the level of BAC. Therefore, we continue using $\gamma = 10^{-3}$ and $\lambda = 10^2$. Table 4 shows the mean and standard deviation (in parentheses) of the estimated parameters based on 1000 series of different lengths simulated using the daily equivalent of the three-state model (6). We do not show results for series of 250 observations, as this is too few to estimate a three-state model with this level of persistence.

As expected, adding an intermediate third state has a very negative effect on the BAC. With three states the difference between

Table 5
Mean and standard deviation (in parentheses) of estimated parameters based on 1000 series of different lengths simulated from conditional *t*-distributions with five degrees of freedom using the daily parameter values. Bold entries identify the best estimate between MLE, spectral, and jump.

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC
250									
True	0.0006	−0.0008	0.0078	0.0174	0.0021	0.0120	0.9802 (0.0750)	0.7968 (0.3200)	0.8885 (0.0917)
MLE$_K$	0.0005 (0.0019)	−0.0005 (0.0121)	0.0071 (0.0046)	0.0153 (0.0056)	0.2041 (0.2018)	0.5197 (0.3428)	0.8600 (0.1949)	0.6690 (0.2947)	0.7645 (0.0955)
MLE	**0.0006** (0.0014)	−0.0003 (0.0065)	0.0072 (0.0044)	**0.0155** (0.0066)	0.1137 (0.1447)	0.4090 (0.3566)	**0.9020** (0.1619)	0.7350 (0.2840)	**0.8185** (0.0835)
Spectral	0.0007 (0.0024)	−**0.0006** (0.0066)	**0.0075** (0.0044)	0.0140 (0.0080)	0.0715 (0.0480)	0.1635 (0.0710)	0.7653 (0.1893)	0.6568 (0.2524)	0.7111 (0.0542)
Jump	0.0005 (0.0024)	−0.0004 (0.0057)	0.0084 (0.0050)	0.0149 (0.0077)	**0.0082** (0.0191)	**0.0421** (0.0492)	0.8778 (0.1840)	**0.7530** (0.3130)	0.8154 (0.0624)
500									
True	0.0006	−0.0008	0.0078	0.0174	0.0021	0.0120	0.9873 (0.0217)	0.8361 (0.2590)	0.9117 (0.0756)
MLE$_K$	**0.0006** (0.0007)	0.0003 (0.0093)	0.0067 (0.0027)	0.0165 (0.0046)	0.1238 (0.1469)	0.4156 (0.3462)	0.9199 (0.1307)	0.7080 (0.2788)	0.8140 (0.1060)
MLE	**0.0006** (0.0006)	−0.0001 (0.0033)	0.0068 (0.0024)	0.0167 (0.0044)	0.0830 (0.1014)	0.3621 (0.3428)	0.9395 (0.1083)	0.7465 (0.2687)	0.8430 (0.0965)
Spectral	0.0007 (0.0016)	−**0.0004** (0.0051)	0.0073 (0.0030)	0.0153 (0.0071)	0.0563 (0.0379)	0.1580 (0.0705)	0.8104 (0.1586)	0.6750 (0.2445)	0.7427 (0.0677)
Jump	**0.0006** (0.0016)	−**0.0004** (0.0049)	**0.0079** (0.0033)	**0.0171** (0.0066)	**0.0043** (0.0087)	**0.0395** (0.0430)	**0.9466** (0.1154)	**0.7869** (0.2608)	**0.8668** (0.0799)
1000									
True	0.0006	−0.0008	0.0078	0.0174	0.0021	0.0120	0.9875 (0.0154)	0.8651 (0.2033)	0.9263 (0.0612)
MLE$_K$	**0.0006** (0.0003)	−**0.0009** (0.0181)	0.0066 (0.0008)	0.0172 (0.0034)	0.0749 (0.0935)	0.3104 (0.3083)	0.9524 (0.0605)	0.7196 (0.2619)	0.8360 (0.1164)
MLE	**0.0006** (0.0003)	−0.0004 (0.0018)	0.0067 (0.0007)	**0.0174** (0.0039)	0.0608 (0.0702)	0.2827 (0.2976)	0.9605 (0.0393)	0.7293 (0.2684)	0.8449 (0.1156)
Spectral	0.0007 (0.0009)	−0.0010 (0.0047)	0.0072 (0.0014)	0.0160 (0.0081)	0.0462 (0.0294)	0.1469 (0.0695)	0.8268 (0.1397)	0.6894 (0.2289)	0.7581 (0.0687)
Jump	**0.0006** (0.0007)	−0.0012 (0.0058)	**0.0078** (0.0012)	0.0188 (0.0097)	**0.0030** (0.0033)	**0.0332** (0.0354)	**0.9769** (0.0585)	**0.7900** (0.2421)	**0.8835** (0.0934)
2000									
True	0.0006	−0.0008	0.0078	0.0174	0.0021	0.0120	0.9892 (0.0091)	0.8958 (0.1264)	0.9425 (0.0467)
MLE$_K$	**0.0006** (0.0002)	0.0000 (0.0117)	0.0067 (0.0004)	**0.0177** (0.0024)	0.0466 (0.0525)	0.2325 (0.2404)	0.9682 (0.0250)	0.7453 (0.2290)	0.8567 (0.1114)
MLE	**0.0006** (0.0002)	−0.0004 (0.0012)	0.0068 (0.0004)	0.0178 (0.0026)	0.0406 (0.0410)	0.2146 (0.2266)	0.9702 (0.0181)	0.7590 (0.2241)	0.8646 (0.1056)
Spectral	0.0008 (0.0005)	−0.0013 (0.0174)	0.0071 (0.0011)	0.0163 (0.0074)	0.0390 (0.0204)	0.1418 (0.0788)	0.8382 (0.1380)	0.7010 (0.1993)	0.7696 (0.0686)
Jump	0.0007 (0.0002)	−**0.0010** (0.0045)	**0.0077** (0.0003)	0.0188 (0.0088)	**0.0025** (0.0010)	**0.0249** (0.0250)	**0.9885** (0.0132)	**0.8234** (0.1712)	**0.9060** (0.0826)

the two initializations of MLE is decisive. When MLE is initialized using *K*-means—similar to how spectral clustering and the jump estimator are initialized—the results are terrible. MLE$_K$ works poorly because the transition probability matrix never converges to something meaningful, which is why the order of the states appears to be wrong despite the presented results being for the most accurate permutation. Interestingly, the BACs of MLE$_K$ and MLE decrease when the simulation length increases.

MLE requires much better initial values to contend with the two other estimators. That is despite having the advantage of knowing the true underlying model and even if 2000 observations are available for estimation. The challenges are most evident when looking at the transition probabilities where, once again, the jump estimates are by far the best. Even for the jump estimator, the BAC is much lower than what is achieved using the true parameter values.

4.3. Misspecified conditional distributions

In order to compare the different estimation methods when conditional distributions are misspecified, we repeat the two-state simulation study, only this time we sample the observations from Student's *t* rather than Gaussian distributions. The conditional distributions have the same mean and standard deviation, but are *t*-distributions with five degrees of freedom, which have heavier tails than the Gaussian distribution. Several studies have found that this is in fact a better fit to the empirical distribution of financial returns (Bulla, 2011; Cont, 2001; Nystrup et al., 2017b).

Table 5 shows the mean and standard deviation (in parentheses) of the estimated parameters based on 1000 series of different lengths simulated from conditional *t*-distributions with five degrees of freedom using the daily parameter values. The accuracy using the true parameter values is based on the Viterbi path under the erroneous assumption that conditional distributions are Gaussian.

Compared to Table 1, the BACs are around five percentage points lower as a result of the misspecified conditionals. When simulating series of 250 observations, the BAC of MLE with good initial values is similar to that of the jump estimator; but when simulating longer series the jump estimator has the highest BAC. MLE$_K$ yields significantly worse estimates than MLE at the shortest simulation lengths, once again emphasizing the method's lack of robustness.

The jump estimates of the transition probabilities stand out compared to the other methods. This is even more evident when looking at the plots of the estimated parameters in Fig. 4. Misspecification of the conditional distributions causes a severe upward bias in the estimated transition probabilities for the other methods, leading to unrealistically rapid switching dynamics. Meanwhile, the jump estimator has no difficulty in estimating the transition probabilities. As a result of the jump estimator being very close to the true values, its BAC is relatively close to what can be achieved using the true parameter values. Spectral clustering has a lower BAC than the other methods, but provides better estimates of the transition probabilities compared to MLE, regardless of the initialization.

5. Trading strategy

Several studies have shown the potential benefit from changing asset allocation depending on regimes inferred from financial returns (see, e.g., Bae et al., 2014; Bulla et al., 2011; Guidolin & Timmermann, 2007; Nystrup et al., 2019; Nystrup et al., 2017a; Nystrup et al., 2015a; Nystrup et al., 2018). The HMM is a popular choice for inferring the hidden state of financial markets, because it can match the tendency of financial markets to change their behavior abruptly and the phenomenon that the new behavior often persists for several periods after a change (Ang & Timmermann, 2012). To illustrate the value of estimating the true persistence of the underlying state process, we consider a simple long–short trading strategy based on the inferred states.

5.1. Simulated data

We first consider a simulation example based on the misspecified two-state HMM from the previous section. We reuse the 1000 series of 1000 observations that were simulated using the daily parameter values, but sampled from conditional *t*-distributions with five degrees of freedom. Long and short positions are implemented for the last 250 observations based on the inferred states. In the state with positive mean and low volatility the strategy is long, and in the other state with negative return and higher volatility the strategy is short the underlying stock index.

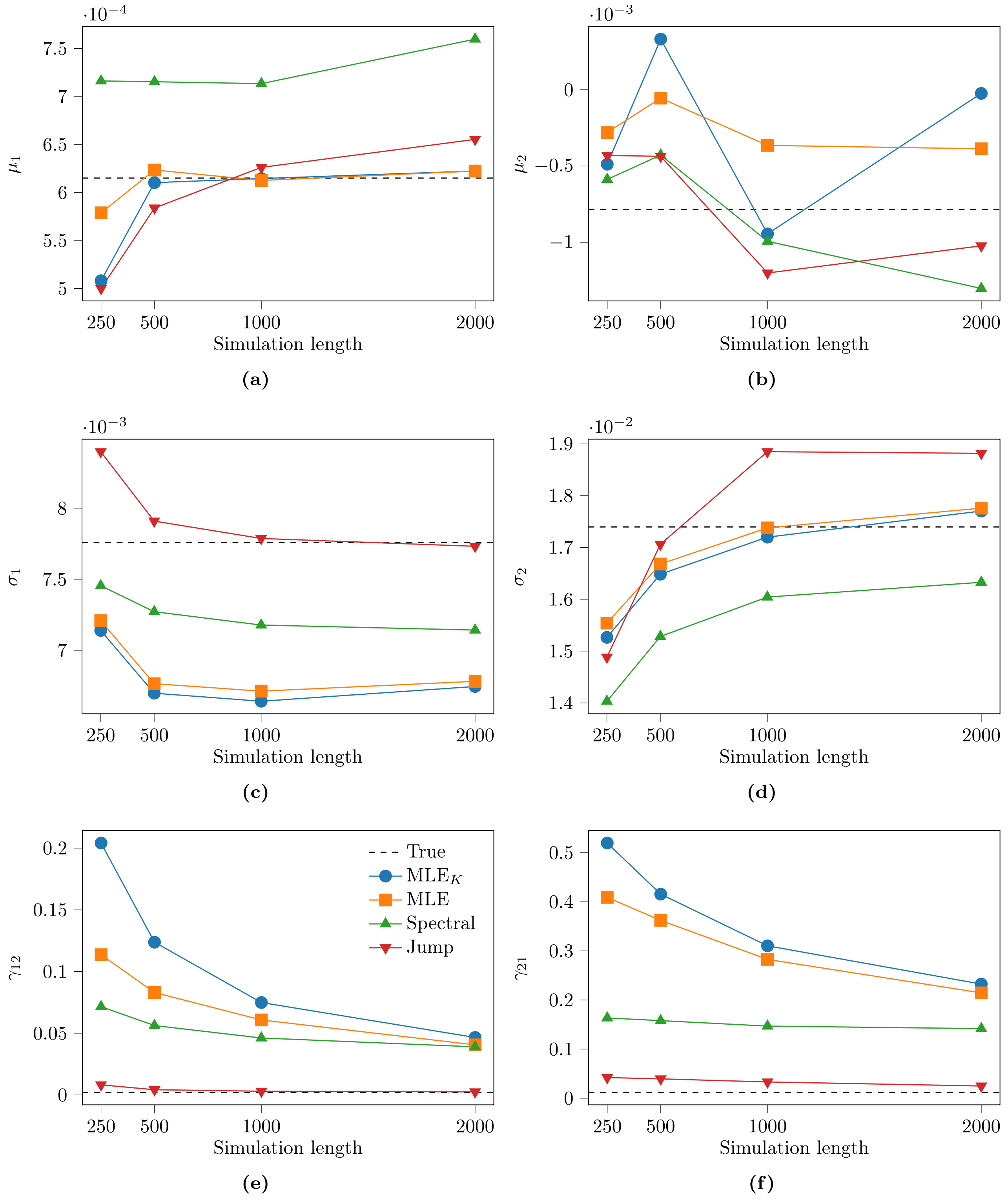

Fig. 4. Estimates of daily parameters based on 1000 simulations from conditional *t*-distributions with five degrees of freedom as a function of the simulation length using different estimation approaches.

Allocating for one year based on three years of data is a realistic setup. We fit the model to the first 750 observations using the different methods and then apply the Viterbi algorithm sequentially to the next 250 observations for state decoding. Thus, differences in the inferred state sequences are entirely due to the differences in the estimated parameter values. An important advantage of spectral clustering and jump estimation is the simultaneous estimation of parameters and state sequence; however, it is outside the scope of this article to extend these methods to streaming applications.

Table 6 shows the results when taking long and short positions based on the states inferred using the different parameter estimates. Results are only shown for MLE as the two initializations yield relatively similar results in this case. Spectral clustering leads

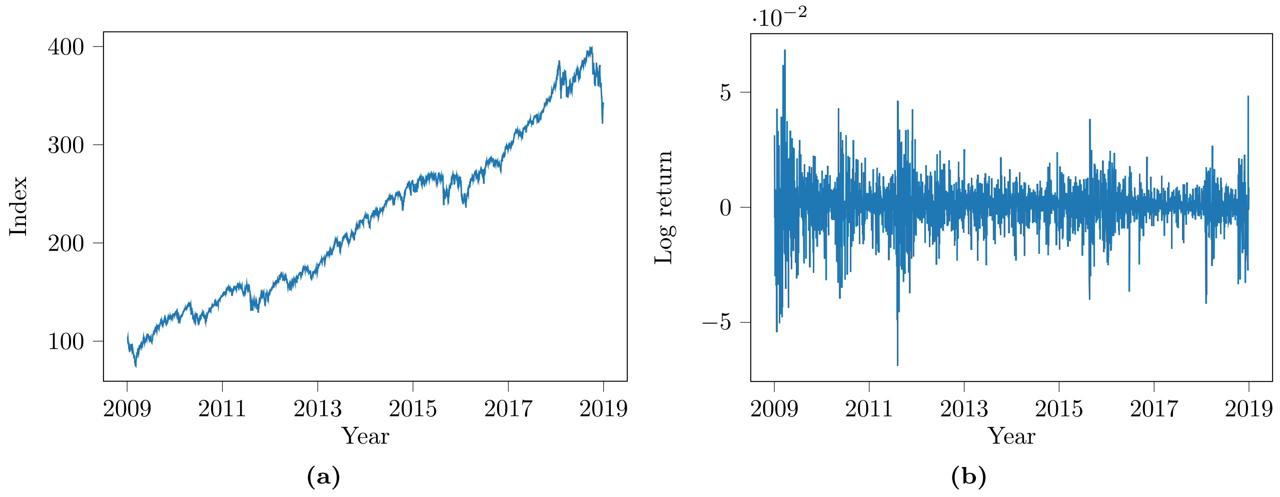

Fig. 5. S&P 500 index values and log returns from 2009 through 2018.

Table 6
Summary of long-short strategy based on 1000 simulations using the last 250 out of 1000 observations simulated from conditional *t*-distributions with five degrees of freedom using the daily parameters. Break-even is the cost per transaction that would offset the return.

	Return	Risk	Sharpe ratio	Turnover	Break-even
MLE	0.1120	0.1767	0.6338	21.52	0.0025
Spectral	0.0899	0.1878	0.4784	17.21	0.0025
Jump	0.1271	0.1836	0.6923	4.55	0.0131

to a lower return with similar risk as the other methods and, consequently, a lower Sharpe ratio—i.e., ratio between return and risk. Risk is measured as the standard deviation of the 1000 annualized returns. The turnover using the spectral parameter estimates is lower than that using the MLE. The two methods have the same break-even transaction cost, which is the cost per transaction that would offset the return.

The long–short strategy based on the states inferred using the jump parameter estimates yields a slightly higher return, risk, and Sharpe ratio than that using the MLE, while incurring a substantially lower turnover. This amounts to a break-even transaction cost that is five times higher (131 basis points), clearly illustrating the value of estimating the true persistence of the underlying state process. Had the simulation been based on series that were shorter than 1000 observations, the difference would have been even bigger.

5.2. Application to S&P 500

Finally, we consider an application to real data using daily values of the S&P 500 total return index from 2009 through 2018. Fig. 5 shows the index and its log returns over the 20-year period. This period after the financial crisis in 2008 is special in that it does not include a major drawdown, which is in sharp contrast to the ten-year period that preceded. As a consequence, it is harder for a regime-based strategy to outperform the index, because it is unable to benefit from prolonged periods of drawdown. This makes it an interesting period to study.

Table 7 shows the estimated parameters for a two-state Gaussian HMM fitted to the log returns of the S&P 500 index. Once again we use $\gamma = 10^{-3}$ and $\lambda = 10^2$, though we could have chosen the values that yield the highest return or Sharpe ratio rather than the highest BAC. Results are only shown for MLE as the two initializations yield almost similar results. The estimated parameters are fairly similar to those used in the simulation study in Section 4.

Table 7
Estimated parameters for a two-state Gaussian HMM fitted to the log returns of the S&P 500 index from 2009 through 2018.

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}
MLE	0.0011	−0.0005	0.0054	0.0154	0.0270	0.0430
Spectral	0.0011	−0.0025	0.0069	0.0200	0.0265	0.1281
Jump	0.0008	−0.0015	0.0092	0.0268	0.0033	0.0182

Table 8
Summary of long–short strategy based on the S&P 500 from 2009 through 2018. Break-even is the cost per transaction that would offset the return.

	Return	Risk	Sharpe ratio	Turnover	Break-even
S&P 500	0.1114	0.1657	0.6726		
MLE	0.1529	0.1656	0.9234	4.175	0.0169
Spectral	0.2181	0.1653	1.319	11.03	0.0089
Jump	0.2370	0.1652	1.434	1.392	0.0736

The higher values for the transition probabilities of MLE and spectral compared to jump could indicate that the conditional distributions have heavier tails than the assumed Gaussian distribution.

Table 8 shows the result when taking long and short positions, respectively, in the S&P 500 index based on the states inferred using the different methods. The purpose is to compare the results using the different estimation methods and not to conclude whether a trading strategy is profitable or not. The latter would require out-of-sample testing. The results confirm the conclusions from the simulation study. Although the turnover of the MLE is lower than in Table 6, it is still above that of the jump estimate, which has by far the highest break-even transaction cost. Over the 20-year period, the Sharpe ratio of the jump estimator compares favorably with that of the S&P 500 index, which is even far above its long-term average.

6. Conclusion

From a theoretical point of view, the MLE has favorable properties; however, it required accurate initial values and correctly specified conditional distributions to obtain those properties in practice. The spectral clustering approach to estimating HMMs had some merit in leading to improved estimates of transition probabilities while being less sensitive to initialization compared to the MLE, but it generally led to lower accuracy when the underlying states were highly persistent. The MLE required considerably better initial values in order to compete with the proposed jump estimate based on temporal features and penalizing jumps. That alone is a strong argument in favor of the jump estimator, since the required prior knowledge is often not available in practical applications.

It was convincing that jump estimation, which required much fewer iterations, could compete with maximizing the correctly specified likelihood function. Even when initialized with sufficiently accurate values, maximizing the likelihood at best yielded the same accuracy as the jump estimator for highly persistent processes. Meanwhile, the jump estimates of transition probabilities were more precise, especially when the samples available for estimation were short. When the hidden state process was less persistent, the jump estimator, as well as spectral clustering, yielded significantly higher accuracy compared to the MLE and, in some cases, even higher than the Viterbi path based on the true parameters.

As the number of latent states increased, the initialization became even more critical to the performance of the MLE. Only if initial values were sufficiently close to the true parameter values, was it able to contend with the jump estimator; but it still led to worse estimates of transition probabilities and oftentimes lower accuracy. When simulating from conditional distributions with heavier tails than the assumed Gaussian distribution, the jump estimator yielded much better results than the other methods considered. The misspecification caused a severe bias in the estimated transition probabilities for the other methods, which led to unrealistically rapid switching dynamics, whereas the jump estimator was not susceptible.

The value of estimating the true persistence of the hidden state process was illustrated through applications to long–short trading strategies. The improved estimates of the latent state process resulted in a substantially lower turnover and higher break-even transaction costs.

Compared to maximization of the likelihood function using the EM algorithm, the advantages of the proposed jump estimator have been shown to include that it learns the hidden state sequence and model parameters simultaneously while providing control over the transition rate, it takes much fewer iterations to converge, it is less sensitive to initial values, it performs better when the number of hidden states increases, and it is more robust to heavy-tailed conditional distributions because it does not assume Gaussianity.

It is left for future research to consider feature selection in depth. The main limitation of the spectral clustering and jump approaches is the forward-looking nature of some of the temporal features used for estimation. This needs to be addressed in future research. At the same time, the possibility of extending the feature space to include additional features that can improve state inference is perhaps the most exciting prospect for future work. This can be easily done because the features do not have to multivariate Gaussian. Moreover, the computational advantage of the jump estimator could be useful in considering data with a frequency higher than daily.

Replacing the forward-looking features is a prerequisite to developing a jump estimator that can be applied in a streaming setting. Developing a streaming implementation is particularly important for the model's application in real-life expert and intelligent systems. Unlike spectral clustering, which happens in the frequency domain, by clustering in the time domain the jump estimator can more easily be extended to a streaming setting. The demonstrated robustness of the jump-penalization approach shows promise for the possibility of extending it to an online estimator.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Credit authorship contribution statement

Peter Nystrup: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Project administration, Resources, Software, Validation, Visualization, Writing - original draft, Writing - review & editing. **Erik Lindström:** Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Supervision, Validation, Writing - review & editing. **Henrik Madsen:** Conceptualization, Funding acquisition, Methodology, Project administration, Resources, Supervision, Validation, Writing - review & editing.

Acknowledgments

This work was supported by Vergstiftelsen and the Centre for IT-Intelligent Energy Systems (CITIES) project funded in part by Innovation Fund Denmark under Grant No. 1305-00027B. We are thankful to Stephen Boyd for introducing us to the framework for fitting jump models, to Kai Zheng for clarifying details about the simulation study in Zheng et al. (2019), and to two reviewers for valuable feedback.

References

Amini, A. A., Chen, A., Bickel, P. J., & Levina, E. (2013). Pseudo-likelihood methods for community detection in large sparse networks. *Annals of Statistics, 41*(4), 2097–2122.
Andersson, S., Rydén, T., & Johansson, R. (2003). Linear optimal prediction and innovations representations of hidden Markov models. *Stochastic Processes and their Applications, 108*(1), 131–149.
Ang, A., & Timmermann, A. (2012). Regime changes and financial markets. *Annual Review of Financial Economics, 4*(1), 313–337.
Arthur, D., & Vassilvitskii, S. (2007). k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM–SIAM symposium on discrete algorithms* (pp. 1027–1035).
Bach, F. R., & Jordan, M. I. (2004). Learning spectral clustering. In *Advances in neural information processing systems* (pp. 305–312).
Bae, G. I., Kim, W. C., & Mulvey, J. M. (2014). Dynamic asset allocation for varied financial markets under regime switching framework. *European Journal of Operational Research, 234*(2), 450–458.
Bakis, R. (1976). Continuous speech recognition via centisecond acoustic states. *Journal of the Acoustical Society of America, 59*(S1), S97.
Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics, 41*(1), 164–171.
Beal, M. J., Ghahramani, Z., & Rasmussen, C. E. (2002). The infinite hidden Markov model. In *Advances in neural information processing systems* (pp. 577–584).
Bellman, R. E. (1957). *Dynamic programming*. Princeton University Press: Princeton.
Bemporad, A., Breschi, V., Piga, D., & Boyd, S. (2018). Fitting jump models. *Automatica, 96*, 11–21.
Bhardwaj, S., Sharma, V., Srivastava, S., Sastry, O., Bandyopadhyay, B., Chandel, S., & Gupta, J. (2013). Estimation of solar radiation using a combination of hidden Markov model and generalized fuzzy model. *Solar Energy, 93*, 43–54.
Blanco, I., & Morales, J. M. (2017). An efficient robust solution to the two-stage stochastic unit commitment problem. *IEEE Transactions on Power Systems, 32*(6), 4477–4488.
Box, G. E. P. (1979). Robustness in the strategy of scientific model building. In R. L. Launer, & G. N. Wilkinson (Eds.), *Robustness in statistics* (pp. 201–236). Academic Press: New York.
Boyd, S., Busseti, E., Diamond, S., Kahn, R. N., Koh, K., Nystrup, P., & Speth, J. (2017). Multi-period trading via convex optimization. *Foundations and Trends in Optimization, 3*(1), 1–76.
Brodersen, K. H., Ong, C. S., Stephan, K. E., & Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *Proceedings of the twentieth international conference on pattern recognition* (pp. 3121–3124).
Bulla, J. (2011). Hidden Markov models with *t* components. Increased persistence and other aspects. *Quantitative Finance, 11*(3), 459–475.
Bulla, J., & Bulla, I. (2006). Stylized facts of financial time series and hidden semi-Markov models. *Computational Statistics & Data Analysis, 51*(4), 2192–2209.
Bulla, J., Mergner, S., Bulla, I., Sesboüé, A., & Chesneau, C. (2011). Markov-switching asset allocation: Do profitable strategies exist? *Journal of Asset Management, 12*(5), 310–321.
Chaudhuri, K., Chung, F., & Tsiatas, A. (2012). Spectral clustering of graphs with general degrees in the extended planted partition model. *Journal of Machine Learning Research, 35*, 1–23.
Choo, K. H., Tong, J. C., & Zhang, L. (2004). Recent applications of hidden Markov models in computational biology. *Genomics, Proteomics & Bioinformatics, 2*(2), 84–96.
Cont, R. (2001). Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance, 1*(2), 223–236.

Davis, R. I. A., & Lovell, B. C. (2004). Comparing and evaluating HMM ensemble training algorithms using train and test and condition number criteria. *Pattern Analysis & Applications, 6*(4).

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological), 39*(1), 1–38.

Dias, J. G., & Ramos, S. B. (2014). Dynamic clustering of energy markets: An extended hidden Markov approach. *Expert Systems with Applications, 41*(17), 7722–7729.

Dias, J. G., Vermunt, J. K., & Ramos, S. B. (2015). Clustering financial time series: New insights from an extended hidden Markov model. *European Journal of Operational Research, 243*(3), 852–864.

Fiecas, M., Franke, J., von Sachs, R., & Kamgaing, J. T. (2017). Shrinkage estimation for multivariate hidden Markov models. *Journal of the American Statistical Association, 112*(517), 424–435.

Fischer, I., & Poland, J. (2005). Amplifying the block matrix structure for spectral clustering. In *Proceedings of the fourteenth annual machine learning conference of belgium and the netherlands* (pp. 21–28).

Fox, E. B., Sudderth, E. B., Jordan, M. I., & Willsky, A. S. (2011). A sticky HDP-HMM with application to speaker diarization. *Annals of Applied Statistics, 5*(2A), 1020–1056.

Gales, M., & Young, S. (2008). The application of hidden Markov models in speech recognition. *Foundations and Trends in Signal Processing, 1*(3), 195–304.

García-Galicia, M., Carsteanu, A. A., & Clempner, J. B. (2019). Continuous-time reinforcement learning approach for portfolio management with time penalization. *Expert Systems with Applications, 129*, 27–36.

Ghahramani, Z., & Jordan, M. I. (1997). Factorial hidden Markov models. *Machine Learning, 29*(2–3), 245–273.

Guidolin, M., & Timmermann, A. (2007). Asset allocation under multivariate regime switching. *Journal of Economic Dynamics and Control, 31*(11), 3503–3544.

Gustafsson, F. (2000). *Adaptive Filtering and Change Detection*. Wiley: West Sussex.

Hallac, D., Nystrup, P., & Boyd, S. (2019). Greedy Gaussian segmentation of multivariate time series. *Advances in Data Analysis and Classification, 13*(3), 727–751.

Hardy, M. R. (2001). A regime-switching model of long-term stock returns. *North American Actuarial Journal, 5*(2), 41–53.

Hsu, D., Kakade, S. M., & Zhang, T. (2012). A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences, 78*(5), 1460–1480.

Huang, X., Acero, A., & Hon, H.-W. (2001). *Spoken language processing: A Guide to theory, algorithm and system development*. Prentice–Hall: New Jersey.

Jia, H., Ding, S., Xu, X., & Nie, R. (2013). The latest research progress on spectral clustering. *Neural Computing and Applications, 24*(7–8), 1477–1486.

Jiang, Y., & Ren, J. (2011). Eigenvector sensitive feature selection for spectral clustering. In D. Gunopulos, T. Hofmann, D. Malerba, & M. Vazirgiannis (Eds.), *Machine learning and knowledge discovery in databases* (pp. 114–129). Springer: Berlin.

Jin, R., Kang, F., & Ding, C. H. (2006). A probabilistic approach for optimizing spectral clustering. In *Advances in neural information processing systems* (pp. 571–578).

Johnson, M. J., & Willsky, A. S. (2013). Bayesian nonparametric hidden semi-Markov models. *Journal of Machine Learning Research, 14*, 673–701.

Joseph, A., & Yu, B. (2016). Impact of regularization on spectral clustering. *Annals of Statistics, 44*(4), 1765–1791.

Kang, M., Ahn, J., & Lee, K. (2018). Opinion mining using ensemble text hidden Markov models for text classification. *Expert Systems with Applications, 94*, 218–227.

Katz, I., & Crammer, K. (2015). Outlier-robust convex segmentation. In *Proceedings of the twenty-ninth AAAI conference on artificial intelligence*.

Kim, S.-J., Koh, K., Boyd, S., & Gorinevsky, D. (2009). ℓ_1 trend filtering. *SIAM Review, 51*(2), 339–360.

Langrock, R., & Zucchini, W. (2011). Hidden Markov models with arbitrary state dwell-time distributions. *Computational Statistics & Data Analysis, 55*(1), 715–724.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory, 28*(2), 129–137.

von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and Computing, 17*(4), 395–416.

Meila, M., & Shi, J. (2001). Learning segmentation by random walks. In *Advances in neural information processing systems* (pp. 873–879).

Nefian, A. V., & Hayes, M. H. (1998). Hidden Markov models for face recognition. In *Proceedings of the 1998 IEEE international conference on acoustics, speech and signal processing: 5* (pp. 2721–2724). IEEE.

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (pp. 849–856).

Nielsen, M. G., Morales, J. M., Zugno, M., Pedersen, T. E., & Madsen, H. (2016). Economic valuation of heat pumps and electric boilers in the Danish energy system. *Applied Energy, 167*, 189–200.

Nystrup, P., Boyd, S., Lindström, E., & Madsen, H. (2019). Multi-period portfolio selection with drawdown control. *Annals of Operations Research, 282*(1–2), 245–271.

Nystrup, P., Hansen, B. W., Larsen, H. O., Madsen, H., & Lindström, E. (2017a). Dynamic allocation or diversification: A regime-based approach to multiple assets. *Journal of Portfolio Management, 44*(2), 62–73.

Nystrup, P., Hansen, B. W., Madsen, H., & Lindström, E. (2015a). Regime-based versus static asset allocation: Letting the data speak. *Journal of Portfolio Management, 42*(1), 103–109.

Nystrup, P., Hansen, B. W., Madsen, H., & Lindström, E. (2016). Detecting change points in VIX and S&P 500: A new approach to dynamic asset allocation. *Journal of Asset Management, 17*(5), 361–374.

Nystrup, P., Madsen, H., & Lindström, E. (2015b). Stylised facts of financial time series and hidden Markov models in continuous time. *Quantitative Finance, 15*(9), 1531–1541.

Nystrup, P., Madsen, H., & Lindström, E. (2017b). Long memory of financial time series and hidden Markov models with time-varying parameters. *Journal of Forecasting, 36*(8), 989–1002.

Nystrup, P., Madsen, H., & Lindström, E. (2018). Dynamic portfolio optimization across hidden market regimes. *Quantitative Finance, 18*(1), 83–95.

Oh, K. J., & Han, I. (2000). Using change-point detection to support artificial neural networks for interest rates forecasting. *Expert Systems with Applications, 19*(2), 105–115.

Petropoulos, A., Chatzis, S. P., & Xanthopoulos, S. (2016). A novel corporate credit rating system based on student's-*t* hidden Markov models. *Expert Systems with Applications, 53*, 87–105.

Petropoulos, A., Chatzis, S. P., & Xanthopoulos, S. (2017). A hidden Markov model with dependence jumps for predictive modeling of multidimensional time-series. *Information Sciences, 412–413*, 50–66.

Pinson, P., Christensen, L., Madsen, H., Sørensen, P., Donovan, M., & Jensen, L. (2008). Regime-switching modelling of the fluctuations of offshore wind generation. *Journal of Wind Engineering and Industrial Aerodynamics, 96*(12), 2327–2347.

Qin, T., & Rohe, K. (2013). Regularized spectral clustering under the degree-corrected stochastic blockmodel. In *Advances in neural information processing systems* (pp. 3120–3128).

Robinson, W. N., & Aria, A. (2018). Sequential fraud detection for prepaid cards using hidden Markov model divergence. *Expert Systems with Applications, 91*, 235–251.

Rousseeuw, K., Émilie Poisson Caillault, Lefebvre, A., & Hamad, D. (2015). Hybrid hidden Markov model for marine environment monitoring. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 8*(1), 204–213.

Rydén, T. (2008). EM versus Markov chain Monte Carlo for estimation of hidden Markov models: A computational perspective. *Bayesian Analysis, 3*(4), 659–688.

Rydén, T., Teräsvirta, T., & Åsbrink, S. (1998). Stylized facts of daily return series and the hidden Markov model. *Journal of Applied Econometrics, 13*(3), 217–244.

Samé, A., Chamroukhi, F., Govaert, G., & Aknin, P. (2011). Model-based clustering and segmentation of time series with changes in regime. *Advances in Data Analysis and Classification, 5*(4), 301–321.

Sedoc, J., Rodu, J., Foster, D., & Ungar, L. (2018). Multiscale hidden Markov models for covariance prediction.

Shamshad, A., Bawadi, M. A., Hussin, W. M. A. W., Majid, T. A., & Sanusi, S. A. M. (2005). First and second order Markov chain models for synthetic generation of wind speed time series. *Energy, 30*(5), 693–708.

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(8), 888–905.

Song, L., Boots, B., Siddiqi, S. M., Gordon, G., & Smola, A. (2010). Hilbert space embeddings of hidden Markov models. In *Proceedings of the twenty-seventh international conference on machine learning* (pp. 991–998).

Verbeek, J. J., Vlassis, N., & Kröse, B. (2003). Efficient greedy learning of Gaussian mixture models. *Neural Computation, 15*(2), 469–485.

Viterbi, A. J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory, 13*(2), 260–269.

Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. In *Proceedings of the seventh IEEE international conference on computer vision: 2* (pp. 975–982). IEEE.

Zelnik-Manor, L., & Perona, P. (2005). Self-tuning spectral clustering. In *Advances in neural information processing systems* (pp. 1601–1608).

Zhang, X., Li, J., & Yu, H. (2011). Local density adaptive similarity measurement for spectral clustering. *Pattern Recognition Letters, 32*(2), 352–358.

Zheng, K., Li, Y., & Xu, W. (2019). Regime switching model estimation: spectral clustering hidden Markov model. *Annals of Operations Research*. doi:10.1007/s10479-019-03140-2.