Continuous Statistical Jump Models for Identifying Financial Regimes

Afşar Onat Aydınhan[1], Petter N. Kolm[2], John M. Mulvey[1], Yizhan Shu[1]

[1]Department of Operations Research and Financial Engineering, Princeton University
[2]Courant Institute of Mathematical Sciences, New York University

August 29, 2023

Abstract

Regime-driven models are popular for addressing temporal patterns in both financial market performance and underlying stylized factors, where a regime describes a period with relatively homogeneous behavior. Recently, statistical jump models have been proposed to learn regimes with high persistence, based on clustering temporal features while explicitly penalizing jumps across regimes. In this article, we extend the jump model by generalizing the discrete hidden state variable into a probability vector over all regimes. This allows us to estimate the probability of being in each regime, providing valuable information for downstream tasks such as regime-aware portfolio models and risk management. Our model's smooth transition from one regime to another enhances robustness over the original discrete model. We provide a probabilistic interpretation of our continuous model and demonstrate its advantages through simulations and real-world data experiments. The interpretation motivates a novel penalty term, called *mode loss*, which pushes the probability estimates to the vertices of the probability simplex thereby improving the model's ability to identify regimes.

Contents

1 Introduction

Regime-switching models have gained significant popularity in economics and finance for their ability to capture the sudden shifts in the dynamics of macroeconomic variables (Hamilton and Susmel, 1994; Stock and Watson, 1996; Hamilton, 2010) and financial market dynamics (Rydén et al., 1998; Bulla, 2011; Ang and Timmermann, 2012). These models can be used to partition time series data into distinct regimes, where each regime represents a period of relatively homogeneous behavior. Regimes can persist for extended periods and often have intuitive interpretations, such as representing different phases of the business cycle (Hamilton, 1989), periods of low or high volatility (Schwert, 1989), and bull or bear markets (Pagan and Sossounov, 2003).

Regime-switching models hold significant appeal in financial applications due to their effective ability to align with changing fundamentals, which often can only be interpreted retrospectively (such as the lagged identification of NBER business cycles). These models enable ex ante real-time forecasting (Ang and Timmermann, 2012), providing valuable insights for decision-making. For instance, by incorporating regime-switching dynamics into the investment process, investors can take advantage of the different characteristics and performance of assets across regimes and make informed decisions based on the anticipated regime changes. A number of studies (Elliott et al., 2010; Bulla et al., 2011; Bae et al., 2014; Mulvey and Liu, 2016; Reus and Mulvey, 2016) document the profitability of employing a regime-aware investment strategy.

Classical *hidden Markov models* (HMMs) exemplify the fundamental nature of regime-switching models. Renowned as a statistical model for sequential data, HMMs have been successfully deployed in various fields including speech recognition (Gales and Young, 2007; Rabiner, 1989), bioinformatics (Durbin et al., 1998), finance (Mamon and Elliott, 2007), and traffic prediction (Qi and Ishak, 2014; Goh et al., 2012). In an HMM, each time series observation depends only on a hidden (unobserved) state variable at the specific time, and is independent of past and future hidden states, given the present one. The dynamics of the hidden state sequence are modeled by a finite-state homogeneous first-order Markov chain, where the value of each state variable is frequently interpreted as the probability of being in a specific regime.

HMMs have found significant applications in financial time series analysis, including the modeling of returns (Hardy, 2001), interest rates (Gray, 1996), high-frequency trading (Cartea and Jaimungal, 2013), and derivatives pricing (Goutte et al., 2017). The presence of multiple hidden states in HMMs allows the marginal distribution of each observation to be a mixture model, thereby providing greater flexibility compared to that of parametric distribution families like the normal- and Student's t-distributions. Temporally, the hidden state sequence's Markovian structure is capable of capturing abrupt jumps and regime persistence to a certain extent. HMMs and their variants have demonstrated the ability to reproduce numerous stylized facts of financial return series, including fat tails, volatility clustering, skewness, and time-varying correlations (Rydén et al., 1998).

However, real-world applications often pose challenges for HMMs, due to model misspecification of both the conditional and sojourn time distributions. Other stylized facts, such as low signal-to-noise ratio, high persistence and imbalance among regimes, typically necessitate a time series with an unrealistically long history for the estimation method to achieve satisfactory statistical accuracy. Such misspecification or misestimation can lead to instability in the inferred hidden state sequence and a lack of true persistence (Bulla, 2011). For example, if an impersistent hidden state sequence is utilized as the basis for portfolio allocation, the frequent transitions between different regimes can result in high turnover, excessive transaction costs, and ultimately inferior performance (Nystrup et al., 2020b).

Several clustering-based alternatives have been proposed in response to the challenges of fitting HMMs as regime-switching models, including *spectral clustering HMMs* (Zheng et al., 2021) and *statistical jump models* (JMs) (Nystrup et al., 2020b,a, 2021). In this article, we introduce the *continuous statistical jump models* (CJMs)[1], an extension of the original jump models where the discrete hidden state variable is replaced by a probability vector over the states. This extension enables us to not only assign a label to the current state but also estimate the probability of each time period belonging to a particular state. This additional information is crucial for various downstream financial applications, such as regime-aware portfolio construction (Li and Mulvey, 2021), risk management (Uysal and Mulvey, 2021), and optimal execution (Li and Mulvey, 2023; Sawhney, 2020).

1.1 Related Works

In time series analysis, it is often valuable to partition the data into consecutive time periods that exhibit similar characteristics. This allows us to fit separate models to each segment, thereby capturing segment- or regime-specific characteristics and their changes more effectively. This task has been extensively studied in various contexts, including cyclical analysis (Bry and Boschan, 1971), change point detection (Aminikhanghahi and Cook, 2017), segmentation (Himberg et al., 2001), mixture models (Picard et al., 2011), and trend filtering (Kim et al., 2009; Gu and Mulvey, 2021). These approaches provide valuable insights into understanding and modeling the underlying dynamics of financial time series.

After the seminal work of Rydén et al. (1998), which fitted simple HMMs on daily financial return series, researchers have proposed numerous extensions to HMMs. By using conditional distributions with heavy tails (Bulla, 2011) and sojourn time distributions other than memoryless geometric distributions (Bulla and Bulla, 2006), the resulting models better capture long memory, exhibit improved persistence of the hidden state sequence, and can reproduce many stylized facts of financial time series. Nystrup et al. (2017) fit classical HMMs, but in an adaptive way that allows for time-varying parameters, resulting in similar improved performance. Other extensions to classical HMMs have been explored, including the utilization of higher-order Markov chains to model the dynamics of the hidden state sequence (Zhang et al., 2019), the incorporation of multi-scale hierarchical structures within HMMs (Fine et al., 1998), and the use of HMMs with distributed state representations (Ghahramani and Jordan, 1995). A significant drawback of classical discrete-time HMMs is the quadratic increase in the number of parameters as the number of states grows. Consequently, researchers are often constrained to utilizing models with only a few states due to this limitation. To address this issue, Nystrup et al. (2015) propose fitting HMMs in continuous time, where the number of parameters increases linearly with the number of states.

While maximum likelihood estimation (MLE) is commonly used to estimate HMMs, gradient-based optimizations of the likelihood function is computationally difficult due to the non-concavity resulting from the integration over the hidden state variables. However, the Baum-Welch algorithm (Baum et al., 1970), an iterative procedure later known to be a specialization of the expectation–maximization (EM) algorithm (Dempster et al., 1977), is only guaranteed to convergence to a local optimum. To address the problem of local optima, Hsu et al. (2012) propose a singular value decomposition (SVD)-based spectral algorithm with provable guarantees, considering that a local optimum may not correspond to the global maximum of the likelihood function (Wu, 1983). This algorithm is specifically designed for HMMs with a large number of hidden states. Bulla and Berzel (2008) propose a hybrid algorithm that combines the benefits of both direct numerical maximization and EM algorithms. Several authors have introduced Bayesian approaches as alternative estimation methods (Rydén, 2008; Ebbers et al., 2017).

Recently, there has been a growing interest in clustering-based methods for estimating regime-switching models as an alternative to the HMM approach. These methods aim to address challenges such as model misspecification, difficulty in optimizing the likelihood function and sensitivity to poor initialization. One such approach is the *spectral clustering HMM* (SC-HMM) proposed by Zheng et al. (2021), which extends the method introduced by Hsu et al. (2012) to HMMs with continuous observations. SC-HMM first constructs a feature set based on local observations and applies spectral clustering (Ng et al., 2001) to group the data into clusters using the spectral analysis of the Laplacian matrix of pairwise feature similarities. Then it

[1]For the sake of brevity, we will omit the term "statistical" going forward. We comment that jump models are not related to jump-diffusion models, a common class of stochastic processes.

estimates the distributional parameters within each cluster and computes the empirical transition matrix. Frequently, the SC-HMM achieves better accuracy and robustness compared to MLE.

Time series segmentation differs from general clustering problems as it incorporates temporal information. The jump model, introduced by Nystrup et al. (2020b), addresses this by applying a clustering algorithm to the set of temporal features in Zheng et al. (2021), while explicitly penalizing jumps between different regimes by a fixed-cost regularization term that is calibrated to achieve a desired level of persistence. The jump penalty is crucial for successfully applying clustering techniques to estimate regime-switching models, as it balances fitting the data with prior assumptions about the persistence of the state sequence. The jump model outperforms MLE and spectral clustering, particularly in cases where the regimes exhibit high persistence. Nystrup et al. (2020a) further leverage the recursive inference proposed in Bemporad et al. (2018), enabling the jump model to be applied in an online fashion. This online capability allows the model to determine which regime a new observation belongs to without the need to parse the historical observation sequence. The model achieves good out-of-sample performance, while not compromising persistence of the hidden state sequence.

The estimation of the jump model is typically performed using a coordinate descent method. To address the issue of potential convergence to local suboptimal solutions, Lin (2023) proposes a *mixed-integer programming* (MIP) based estimation approach. This method effectively mitigates the issue of local optima and allows for the integration of additional constraints, thereby improving the estimation of jump models.

Several extensions have been developed to utilize regime-switching models in high-dimensional feature spaces for efficient dimensionality reduction. Nystrup et al. (2021) propose the *sparse statistical jump model* (SJM), which integrates the framework of feature selection in clustering developed by Witten and Tibshirani (2010) into the existing jump model, for simultaneously performing feature selection, parameter estimation, and state classification. To facilitate hyperparameter tuning and model selection in high-dimensional settings, Cortese et al. (2023b) extend the *generalized information criteria* (GIC) for high-dimensional penalized models to sparse statistical jump models. Cortese et al. (2023a) apply this technique to investigate what drives cryptocurrency returns. Furthermore, Sawhney (2020) explores feature extraction and dimension reduction within a regime-switching model for optimal execution using *variational auto-encoders* (VAEs).

1.2 Contributions

In this article we make several contributions to the literature on statistical jump models and their empirical application to financial time series.

First, we introduce a continuous extension of the original jump model by allowing the discrete hidden state variable to be a probability vector over the regimes. We propose a jump penalty term for the probability vectors of the continuous model and establish its connection to the jump penalty in the discrete model through optimal transport. The jump penalty ensures high persistence in the estimated probability vector, enabling the model to capture the true persistence in the underlying regime-switching process. Additionally, the specific mathematical formulation of the jump penalty term leads to a computationally tractable optimization problem used in estimating the new continuous statistical jump model.

Second, we provide a probabilistic interpretation of the continuous jump model under the assumption that the data is generated from a generalized HMM governed by a Markov process defined on a probability simplex. We derive the transition kernel of this Markov process from the jump penalty, capturing the transition probabilities across different regimes. Notably, this interpretation suggests an additional penalty term referred to as the *mode loss* (see, Bemporad et al. (2018) for a discussion) that promotes sparsity of the hidden probability vector, thereby increasing the model's ability to provide definitive state assignments.

Third, we conduct an extensive simulation study to evaluate the performance of the continuous jump model in different settings, including two- and three state models with different levels of persistence, and misspecified conditional distributions and sojourn times. In our simulations, the continuous jump model outperforms the HMM estimated using the Baum-Welch algorithm and the discrete jump mode across most settings, both in terms of state classification and parameter estimation. In particular, the significant improvement in the area under the Receiver Operating Characteristic (ROC) curve demonstrates that the continuous model provides more reliable probability estimates, indicating an ability to better capture the true underlying probabilities and thereby make more accurate predictions.

The continuous jump model offers several advantages over the discrete model that are important in financial applications. It provides estimated probabilities, which are required for tasks like portfolio construction and risk management. The smooth transition in probability estimation, as opposed to abrupt jumps between regimes, improves the model's robustness to model misspecification. The continuous model is less sensitive to hyperparameter tuning and is more accurate in detecting rebounds during bear market periods.

This article is organized as follows. We review the original jump model in Section 2. In Section 3, we introduce the new continuous jump model, analyze its properties, and provide a probabilistic interpretation. Then in Section 4 we present an extensive empirical simulation study. We provide an application of the new model by conducting an analysis of the economic regimes of the Nasdaq Index across two distinct time periods in Section 5. Finally, Section 6 concludes.

2 Statistical Jump Models

The statistical jump model framework proposed in Bemporad et al. (2018) incorporates temporal information when fitting multiple models to a time series. Specifically, given an observation sequence of D (standardized) features $\boldsymbol{Y} := \{\boldsymbol{y}_0, \dots, \boldsymbol{y}_{T-1}\}$ with $\boldsymbol{y}_t \in \mathbb{R}^D$ for all t, we estimate a statistical jump model with K states by solving the optimization problem

$$\arg\min_{\Theta, \boldsymbol{S}} \sum_{t=0}^{T-1} l(\boldsymbol{y}_t, \boldsymbol{\theta}_{s_t}) + \lambda \sum_{t=1}^{T-1} \mathbb{1}_{\{s_{t-1} \neq s_t\}}, \tag{1}$$

where $\Theta := \left\{\boldsymbol{\theta}_k \in \mathbb{R}^D : k = 0, \dots, K-1\right\}$ are the model parameters with $\boldsymbol{\theta}_k \in \mathbb{R}^D$ for all k, $\boldsymbol{S} := \{s_0, \dots, s_{T-1}\}$ denotes the state sequence, and $\lambda \in \mathbb{R}_+$ is a hyperparameter referred to as the jump penalty. We emphasize that each hidden state s_t takes on values in $\{0, \dots, K-1\}$ and is associated with a state-specific model parameter $\boldsymbol{\theta}_k$. At a high level, we can intuitively understand the objective function (1) as a compromise between fitting the data using multiple models and incorporating our prior beliefs about the persistence of the state sequence. After estimating a jump model, we can infer the transition probability matrix based on the optimal hidden state sequence.

Throughout this article, we employ the scaled squared ℓ_2-distance as the loss function, defined by $l(\boldsymbol{y}, \boldsymbol{\theta}) := \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{\theta}\|_2^2$. A general quadratic form for the loss function, e.g. squared Mahalanobis distance, is also applicable if we have prior knowledge of the covariance matrix of the features or if we intend to learn it from the data (Hallac et al., 2017).

2.1 Features

For financial return series of an individual instrument, the observation sequence consists of scalar values $x_0, \dots, x_{T-1} \in \mathbb{R}$. In order to enhance model performance, we construct a set of additional features. Zheng et al. (2021) propose a feature set based on local observations, including rolling means and volatilities calculated over different window sizes. An important characteristic of their feature set is that it incorporates both backward- and forward-looking components, allowing for improved performance through smoothing. However, if the fitted jump model is intended for use in online prediction tasks, where the state of a new observation needs to be identified as soon as it arrives, only backward-looking features are applicable. In our article, we adopt the backward-looking features proposed by Nystrup et al. (2020a) and summarized in Algorithm 1. In particular, we choose two window lengths, $w = 6, 14$, resulting in a total of fifteen standardized features. It is worth mentioning that the features can also come from exogenous time series, such as macro-economic variables, or financial time series of other instruments (Dias et al., 2015; Cortese et al., 2023a).

2.2 Model Fitting

Nystrup et al. (2020b) propose to minimize the objective function in equation (1) using a coordinate descent algorithm, which alternates between (i) minimizing the model parameters with a fixed state sequence,

```
Algorithm 1: Generate features for the jump models
  Input: Univariate time series $\{x_0, \dots, x_{T-1}\}$ and window lengths $w = 6, 14$.
  Compute features at each time $t$:
    1. Observation: $x_t$.
    2. Absolute change: $|x_t - x_{t-1}|$.
    3. Previous absolute change: $|x_{t-1} - x_{t-2}|$.
  Compute features at each time $t$ for $w = 6, 14$:
    4. Centered mean: $\mathrm{mean}[x_{t-w+1}, \dots, x_t]$.
    5. Centered standard deviation: $\mathrm{std}[x_{t-w+1}, \dots, x_t]$.
    6. Left mean: $\mathrm{mean}[x_{t-w+1}, \dots, x_{t-\frac{w}{2}}]$.
    7. Left standard deviation: $\mathrm{std}[x_{t-w+1}, \dots, x_{t-\frac{w}{2}}]$.
    8. Right mean: $\mathrm{mean}[x_{t-\frac{w}{2}+1}, \dots, x_t]$.
    9. Right standard deviation: $\mathrm{std}[x_{t-\frac{w}{2}+1}, \dots, x_t]$.
  Output: Fifteen-dimensional feature set.
```

and (ii) minimizing the state sequence with fixed model parameters. We summarize the coordinate descent algorithm in Algorithm 2. The iteration of the algorithm stops either when the state sequence does not change or when the improvement of the value of the objective function is smaller than a specified tolerance. The fitting of model parameters is performed within each cluster, and is computationally tractable when the loss function l is convex with respect to model parameter $\boldsymbol{\theta}$, often leading to an analytical solution. For fixed model parameters, one obtains the optimal state sequence using a dynamic programming (DP) algorithm outlined in Algorithm 3, similar to the Viterbi algorithm (Viterbi, 1967) for obtaining the maximum a posteriori (MAP) estimate of the most likely hidden state sequence in an HMM.

Given the non-convex nature of the objective function in equation (1), we can only expect the coordinate descent algorithm to converge to a stationary point, without any guarantee of finding the global optimum. Wright (2015) examines the use, implementation, and convergence of coordinate descent algorithms for convex objective functions. Notably, global convergence results for non-convex objective functions are only obtainable in particular special cases (see, for example, Bertsekas (1999, Proposition 2.7.1); Attouch et al. (2010); Bolte et al. (2014)). To address the issue of local optima, we run the algorithm ten times with different starting points, generated by the K-means++ algorithm (Arthur and Vassilvitskii, 2007), and retain the solution that obtains the best value of the objective function.

```
Algorithm 2: Coordinate descent algorithm for the discrete jump model
  Input: Time series $\boldsymbol{Y} := \{\boldsymbol{y}_0, \dots, \boldsymbol{y}_{T-1}\}$, number of states $K$, and jump penalty $\lambda \in \mathbb{R}_+$.
  Iterate for $i = 1, \dots$
    (a) Fit model $\Theta^i = \arg\min_\Theta \sum_{t=0}^{T-1} l\left(\boldsymbol{y}_t, \boldsymbol{\theta}_{s_t^{i-1}}\right)$.
    (b) Fit state sequence $\boldsymbol{S}^i = \arg\min_{\boldsymbol{S}} \left\{\sum_{t=0}^{T-1} l(\boldsymbol{y}_t, \boldsymbol{\theta}^i_{s_t}) + \lambda \sum_{t=1}^{T-1} \mathbb{1}_{\{s_{t-1} \neq s_t\}}\right\}$.
  Until $\boldsymbol{S}^i = \boldsymbol{S}^{i-1}$, or the improvement of the value of the objective function is smaller than a
  specified tolerance.
  Compute the empirical transition probability matrix.
  Output: Model parameters and hidden state sequence
```

The time complexity of the DP algorithm is $O(TK^2)$, the same as one forward-backward pass in the E-step of the Baum-Welch algorithm. As we discuss later in Section 3.2, fitting the state sequence with fixed model parameters in the discrete jump model can also be done by solving a relaxed linear programming (LP)

Algorithm 3: Fit the state sequence for the discrete jump model using dynamic programming

```
Input: Loss matrix $\boldsymbol{L} = (l_{t,k}) \in \mathbb{R}^{T \times K}$, where each $l_{t,k} := l(\boldsymbol{y}_t, \boldsymbol{\theta}_k)$, and jump penalty $\lambda \in \mathbb{R}_+$.
Initialize: $V(0, k) = l_{0,k}, \quad k = 0, \dots, K-1$.
for $t = 1, \dots, T-1$ do
    $V(t,k) = l_{t,k} + \min_j \left( V(t-1, j) + \lambda \mathbb{1}_{\{j \neq k\}} \right), \quad k = 0, \dots, K-1.$    (2)
end
Compute optimal value $\hat{v} = \min_k V(T-1, k)$.
Compute optimal state $\hat{s}_{T-1} = \arg\min_k V(T-1, k)$.
for $t = T-1, \dots, 1$ do
    $\hat{s}_{t-1} = \min_j \left( V(t-1, j) + \lambda \mathbb{1}_{\{j \neq \hat{s}_t\}} \right).$    (3)
end
Output: Optimal value $\hat{v}$ and optimal state sequence $\{\hat{s}_0, \dots, \hat{s}_{T-1}\}$.
```

problem. Based on our experience, the DP solution is generally faster than calling an LP solver, especially when the number of states K is relatively low (less than ten).

2.3 Jump Penalty

The hyperparameter λ serves as a control parameter for the fixed-cost regularization term associated with transitions between different states. Its value reflects our prior assumptions about the frequency of state transitions. When λ is set to zero and the loss function l corresponds to the squared ℓ_2-distance, the jump model becomes the K-means algorithm (Lloyd, 1982) which does not take the temporal order into account. As we increase the value of λ, the number of state transitions decreases. Eventually, a single state transition becomes so costly that the jump model is reduced to a single model fitted to the entire time series.

The penalization of jumps is a critical aspect of successfully applying this type of temporal clustering algorithms to estimate regime-switching models. Techniques for selecting the optimal hyperparameter λ include cross-validation methods tailored to the specific application and information criteria (Cortese et al., 2023b). While the objective function in equation (1) applies the same penalty to all jumps between states, it is possible to introduce state-specific penalties for different types of jumps, such as distinguishing between a jump from a bull to a bear market regime and vice versa. To incorporate these penalties, we can define a penalty matrix $\Lambda \in \mathbb{R}_+^{K \times K}$, where each entry $\Lambda_{ij} \geq 0$ represents the penalty for transitioning from state i to state j. This penalty matrix allows for a more nuanced treatment of different types of state transitions.

3 Continuous Statistical Jump Models

In many financial applications, such as portfolio optimization and risk management, having the ability to estimate the probability of each time period belonging to a specific regime is crucial. A probabilistic perspective provides more informative and valuable insights compared to the hard clustering approach in the discrete jump model that assigns a single label to each time period indicating the identified regime. A hard clustering approach lacks the ability to capture the uncertainty associated with regime assignment and the varying probabilities of belonging to different regimes. For this purpose, we propose extending the hidden state space from the discrete set $\{0, \dots, K-1\}$ to the probability simplex $\Delta_{K-1} := \left\{ \boldsymbol{s} = (s_0, \dots, s_{K-1})^\mathsf{T} \in \mathbb{R}^K : \sum_{k=0}^{K-1} s_k = 1, \ \boldsymbol{s} \geq 0 \right\}$. Then we let each hidden state vector $\boldsymbol{s}_t$ be a probability vector over all the states. We use $K-1$ as the subscript of the simplex since its affine dimension is $K-1$. The hidden vector sequence is now represented by $\boldsymbol{S} := \left[\boldsymbol{s}_0, \cdots, \boldsymbol{s}_{T-1} \right]^\mathsf{T} \in \mathbb{R}^{T \times K}$, which should satisfy the constraint

$$\boldsymbol{S} \geq \boldsymbol{0}, \quad \boldsymbol{S}\boldsymbol{1}_K = \boldsymbol{1}_T, \tag{4}$$

where $\boldsymbol{1}_n$ represents a vector of all ones of length n.

Given an observation sequence $\boldsymbol{Y}$ of (standardized) features, we propose solving the following minimization problem

$$\underset{\Theta,\boldsymbol{S}}{\arg\min} \sum_{t=0}^{T-1} L(\boldsymbol{y}_t,\Theta,\boldsymbol{s}_t) + \frac{\lambda}{4}\sum_{t=1}^{T-1} \|\boldsymbol{s}_{t-1}-\boldsymbol{s}_t\|_1^2 \,, \tag{5}$$

where Θ represents the model parameters, $\boldsymbol{S}$ is the hidden vector sequence, and the loss function L is defined as a weighted average of the loss between $\boldsymbol{y}_t$ to each cluster center, i.e.

$$L(\boldsymbol{y}_t,\Theta,\boldsymbol{s}_t) = \sum_k s_{t,k} l(\boldsymbol{y}_t,\boldsymbol{\theta}_k)\,, \tag{6}$$

with l and $s_{t,k}$ denoting the scaled squared ℓ_2-distance and the k-th element of $\boldsymbol{s}_t$, respectively.

We remark that the factor of $1/4$ in the second term of equation (5) ensures that λ is consistent with that of the discrete model, if we were to reduce a continuous model to a discrete one. Like the original jump model, the objective (5) fits the data to multiple models while simultaneously keeping the total number of jumps small. We emphasize that the second term of the the objective function penalizes transitions between two subsequent probability vectors using the squared ℓ_1-norm of their difference. The ℓ_1-norm promotes sparsity in the difference between two consecutive estimated probability vectors, hence encouraging persistence in the inferred state process. In Section 3.2, we provide theoretical justification for the choice of this regularization term for jumps on the probability simplex.

3.1 Model Estimation

We propose to solve the optimization problem in (5) above by a coordinate descent method outlined in Algorithm 4, which alternates between fitting (i) the model parameters, and (ii) the hidden vector sequence, while keeping the other fixed. Specifically, in fitting the model parameters, we solve a weighted minimization problem for each regime, given by

$$\underset{\boldsymbol{\theta}_k}{\arg\min} \sum_{t=0}^{T-1} s_{t,k} l(\boldsymbol{y}_t,\boldsymbol{\theta}_k)\,. \tag{7}$$

Similarly, when the loss function l is convex with respect to the parameter $\boldsymbol{\theta}_k$, this optimization problem is tractable. In the case of scaled squared ℓ_2-distance, we obtain the solution analytically.

When fitting the hidden vector sequence, with the *loss matrix* $\boldsymbol{L} \in \mathbb{R}^{T\times K}$, where $\boldsymbol{L}_{t,k} := (l(\boldsymbol{y}_t,\boldsymbol{\theta}_k))$, as the input, the optimization problem over the state sequence is given by

$$\underset{\boldsymbol{S}:\,\boldsymbol{S}\geq 0,\ \boldsymbol{S}\boldsymbol{1}_K=\boldsymbol{1}_T}{\arg\min} \langle \boldsymbol{S},\boldsymbol{L}\rangle + \frac{\lambda}{4}\sum_{t=1}^{T-1} \|\boldsymbol{s}_{t-1}-\boldsymbol{s}_t\|_1^2 \,, \tag{8}$$

where $\langle \boldsymbol{S},\boldsymbol{L}\rangle := \sum_{t,k} s_{t,k} l_{t,k}$ denotes the inner product between two matrices. As the problem has a quadratic objective function and linear constraints, it is a quadratic programming (QP) problem. Although this optimization problem can be solved with a QP solver, we propose an approximate solution using a DP algorithm, which has a similar structure to the one used in the discrete jump model and in general offers a shorter running time. In particular, we first discretize the probability simplex. In our implementation, we generate a set of uniformly sampled discretized vectors $\boldsymbol{c}_0,\ldots,\boldsymbol{c}_{N-1} \in \Delta_{K-1}$ on the probability simplex with a fixed grid size of $\delta > 0$, and stack them into a matrix $\boldsymbol{C} := \begin{bmatrix}\boldsymbol{c}_0 & \cdots & \boldsymbol{c}_{N-1}\end{bmatrix} \in \mathbb{R}^{K\times N}$. Note that the total number N of candidate vectors scales exponentially with K, i.e. $N = O\left(\lceil \frac{1}{\delta}\rceil^K\right)$. In contrast to the discrete model, which selects one of the K states at each time step t, the continuous model chooses one of the N candidate probability vectors. Furthermore, instead of uniformly penalizing jumps across states with a penalty term λ, the change between two consecutive estimated probability vectors is penalized using the squared ℓ_1-norm of the vector difference. We summarize this algorithm in Algorithm 5. To address the issue of local optima, just like for the discrete jump model we run the algorithm ten times with different starting points, generated by the K-means++ algorithm.

Fitting the hidden vector sequence with the approximate DP algorithm results in a time complexity of $O(TN^2)$ for each iteration. However, as N scales exponentially with K, we need to increase the grid size when the number of states becomes larger. To maintain satisfactory speed, we recommend using fewer than

Algorithm 4: Coordinate descent algorithm for the continuous jump model

```
**Input:** Time series $\boldsymbol{Y} := \{\boldsymbol{y}_0, \ldots, \boldsymbol{y}_{T-1}\}$, number of states $K$, and jump penalty $\lambda \in \mathbb{R}_+$.
Iterate for $i = 1, \ldots$
  (a) Fit model $\Theta^i = \arg\min_\Theta \sum_{t=0}^{T-1} L\left(\boldsymbol{y}_t, \Theta, \boldsymbol{s}_t^{i-1}\right)$.
  (b) Fit state sequence $\boldsymbol{S}^i = \arg\min_{\boldsymbol{S}} \left\{\sum_{t=0}^{T-1} L(\boldsymbol{y}_t, \Theta^i, \boldsymbol{s}_t) + \frac{\lambda}{4} \sum_{t=1}^{T-1} \|\boldsymbol{s}_{t-1} - \boldsymbol{s}_t\|_1^2\right\}$.
Until $\boldsymbol{S}^i = \boldsymbol{S}^{i-1}$, or the improvement of the value of the objective function is smaller than a specified tolerance.
Compute the empirical transition probability matrix.
**Output:** Model parameters and hidden state sequence
```

Algorithm 5: Fit the state sequence for the continuous jump model using dynamic programming

```
**Input:** Loss matrix $\boldsymbol{L} = (l(\boldsymbol{y}_t, \boldsymbol{\theta}_k)) \in \mathbb{R}^{T \times K}$, candidate vectors $\boldsymbol{C} \in \mathbb{R}^{K \times N}$, and jump parameter $\lambda \in \mathbb{R}_+$.
Compute the new loss matrix $\widetilde{L} = (\widetilde{l}_{t,i}) := \boldsymbol{LC} \in \mathbb{R}^{T \times N}$.
Compute the jump penalty matrix $\Lambda = (\Lambda_{i,j}) \in \mathbb{R}^{N \times N}$, where $\Lambda_{i,j} := \frac{\lambda}{4} \|\boldsymbol{c}_i - \boldsymbol{c}_j\|_1^2$.
Initialize: $\widetilde{V}(0, i) = \widetilde{l}_{0,i}, \quad i = 0, \ldots, N-1$.
**for** $t = 1, \ldots, T-1$ **do**
    $\widetilde{V}(t, i) = \widetilde{l}_{t,i} + \min_{j \le N-1} \left(\widetilde{V}(t-1, j) + \Lambda_{j,i}\right), \quad i = 0, \ldots, N-1.$   (9)
**end**
Compute optimal value $\hat{v} = \min_{i \le N-1} \widetilde{V}(T-1, i)$.
Compute optimal index $\hat{i}_{T-1} = \arg\min_{i \le N-1} \widetilde{V}(T-1, i)$, and optimal hidden vector $\hat{\boldsymbol{s}}_{T-1} = \boldsymbol{c}_{\hat{i}_{T-1}}$.
**for** $t = T-1, \ldots, 1$ **do**
    $\hat{i}_{t-1} = \arg\min_{j \le N-1} \left(\widetilde{V}(t-1, j) + \Lambda_{j,\hat{i}_t}\right), \quad \hat{\boldsymbol{s}}_{t-1} = \boldsymbol{c}_{\hat{i}_{t-1}}.$   (10)
**end**
**Output:** Optimal value $\hat{v}$ and optimal hidden vector sequence $\{\hat{\boldsymbol{s}}_0, \ldots, \hat{\boldsymbol{s}}_{T-1}\}$.
```

several hundreds of candidate probability vectors, as the approximate DP algorithm needs to be run several times until convergence. In case the number of states becomes really large, it is preferred to directly call a QP solver. Based on our empirical work, we find that the error introduced by the DP approximation is negligible when the grid size is less than approximately 0.05.

3.2 Jump Penalty for Probability Vectors

In this section we justify the use of *squared ℓ_1-norm of vector difference* used in equation (5) for the jump penalty on the probability simplex. First, our approach uses the ℓ_1-norm to measure the difference between two probability vectors. This choice can be interpreted as the *Wasserstein distance* induced by the discrete metric in the discrete model. Specifically, the jump penalty $\mathcal{L}^{\text{jump}}(k,j) := \lambda \mathbb{1}_{\{k \neq j\}}$, $k, j \in \{0, \ldots, K-1\}$ defines the *discrete metric* on the space $\{0, \ldots, K-1\}$ (Munkres, 2000). A natural way to lift this metric to the probability simplex is via *optimal transport*, where the cost of moving one unit of resources between two points is some power of their distance. The resulting distance between probability vectors is the Wasserstein distance. Remarkably, under the ground metric $\mathcal{L}^{\text{jump}}$, the Wasserstein distance is exactly the scaled ℓ_1-norm (Peyré and Cuturi, 2019, Remark 2.26), also known as the *total variation distance* (Levin et al., 2017, Proposition 4.2), defined by

$$\mathrm{TV}(\boldsymbol{p}, \boldsymbol{q}) := \frac{1}{2} \left\| \boldsymbol{p} - \boldsymbol{q} \right\|_1, \quad \boldsymbol{p}, \boldsymbol{q} \in \Delta_{K-1}. \tag{11}$$

Therefore, the use of the scaled ℓ_1-norm as a distance on the probability simplex in the continuous jump model naturally inherits the discrete metric on $\{0, \ldots, K-1\}$ implicit in the discrete jump model. Furthermore, as mentioned in Section 2.3, when considering a state-specific penalty matrix $\Lambda \in \mathbb{R}_+^{K \times K}$ in the discrete model, the use of the Wasserstein distance with the ground cost matrix chosen as Λ provides a natural choice for the corresponding jump penalty on the probability simplex. This connection allows for a consistent formulation of the penalty terms across both models.

Secondly, however, if we directly use the scaled ℓ_1-norm itself as the jump penalty in (5), the continuous model will be reduced to exactly a discrete jump model. The reason is that, when fitting the hidden vector sequence with fixed model parameters, the optimization problem becomes an LP, namely

$$\underset{\boldsymbol{S}:\, \boldsymbol{S} \geq 0,\ \boldsymbol{S}\mathbf{1}_K = \mathbf{1}_T}{\arg\min} \ \langle \boldsymbol{S}, \boldsymbol{L} \rangle + \frac{\lambda}{2} \sum_{t=1}^{T-1} \left\| \boldsymbol{s}_{t-1} - \boldsymbol{s}_t \right\|_1. \tag{12}$$

We remark that in contrast to (8), in the formula above we apply a factor of $\frac{\lambda}{2}$ to the ℓ_1-norm to make it consistent with the discrete jump model. As per the fundamental theorem of linear programming (Bertsimas and Tsitsiklis, 1997, Theorem 2.7), the minimum of (12) is attained at an extreme point of the feasible set, which corresponds to a corner of the simplex. At the corners, the probability estimation for a specific state is either 0 or 1, indicating a definitive assignment to a single state. Consequently, the optimal hidden vector sequence represents a hard clustering of the observations, reducing the model to a discrete one.[2]

As a result, we choose to use the squared ℓ_1-norm of the vector difference as the final jump penalty on the simplex Δ_{K-1}, which transforms (12) into the QP problem (8). When K is relatively small and the number of candidate probability vectors is limited to a few hundred, we can expedite computations by approximately solving this QP with dynamic programming using Algorithm 5. However, when the number of states are larger, it is preferable to call a QP solver directly.

Using other powers of the ℓ_1-norm of the vector difference, i.e. $\mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t) \propto \left\| \boldsymbol{s}_{t-1} - \boldsymbol{s}_t \right\|_1^{\alpha}$, is also a valid option. This introduces a *mixed norm* penalty (Kowalski, 2009), allowing for the promotion of structured sparsity. However, in our empirical experiments, we did not find significant differences when using α strictly greater than two. Therefore, we use $\alpha = 2$ in our empirical work in this article to maintain the QP form.

3.3 Probabilistic Interpretation

In this section, we establish a probabilistic interpretation of the continuous jump model. Our aim is to provide a framework for understanding the model's behavior based on probabilistic and statistical principles.

[2]We remark that the LP problem (12) is a tight relaxation of the dynamic programming problem for determining the hidden state sequence in the discrete jump model (1) when the other model parameters are held fixed.

Unlike traditional HMMs where the state sequence $\boldsymbol{S}$ follows a finite-state Markov chain, in the continuous jump model it follows a Markov process on the probability simplex.

We assume that the sequence $(\boldsymbol{Y}, \boldsymbol{S})$ is generated from a *generalized hidden Markov model* (GHMM), where the emission distribution of $\boldsymbol{y}_t$ is a mixture model and the hidden state $\boldsymbol{s}_t$ represents the prior probabilities of the observation belonging to each mixture component. Formally, we make the following assumptions:

1. (Hidden Markov process) The hidden state sequence $\boldsymbol{S}$ follows a Markov process on the probability simplex, with initial distribution $\pi : \Delta_{K-1} \to \mathbb{R}_+$, and transition kernel $K : \Delta_{K-1} \times \Delta_{K-1} \to \mathbb{R}_+$, i.e.
$$p(\boldsymbol{S}) = \pi(\boldsymbol{s}_0) \times \prod_{t=1}^{T-1} K(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t)\,. \tag{13}$$

2. (The conditional likelihood of $\boldsymbol{Y}$) Given the hidden state sequence $\boldsymbol{S}$, the observation $\boldsymbol{y}_t$ only depends on $\boldsymbol{s}_t$, i.e.
$$p(\boldsymbol{Y}|\boldsymbol{S}, \Theta) = \prod_{t=0}^{T-1} p(\boldsymbol{y}_t|\Theta, \boldsymbol{s}_t)\,, \tag{14}$$
where $p(\cdot|\Theta, \boldsymbol{s})$ is the emission distribution specified next.

3. (Emission distribution) $p(\cdot|\Theta, \boldsymbol{s})$ follows a mixture model, where all of the K components belong to the same parametric family $\{p(\cdot|\boldsymbol{\theta}) : \boldsymbol{\theta} \in \mathbb{R}^D\}$. The k-th component has parameter $\boldsymbol{\theta}_k$ and weight s_k, i.e.
$$p(\boldsymbol{y}|\Theta, \boldsymbol{s}) = \sum_{k=0}^{K-1} s_k p(\boldsymbol{y}|\boldsymbol{\theta}_k)\,. \tag{15}$$

We recall that the objective function of the continuous jump model takes the form
$$J(\Theta, \boldsymbol{S}) = \sum_{t=0}^{T-1} L(\boldsymbol{y}_t, \Theta, \boldsymbol{s}_t) + \mathcal{L}^{\text{init}}(\boldsymbol{s}_0) + \sum_{t=1}^{T-1} \mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t)\,, \tag{16}$$
where $L(\boldsymbol{y}_t, \Theta, \boldsymbol{s}_t) = \sum_k s_{t,k} l(\boldsymbol{y}_t, \boldsymbol{\theta}_k)$. We denote by $\mathcal{L}(\boldsymbol{S}) := \mathcal{L}^{\text{init}}(\boldsymbol{s}_0) + \sum_{t=1}^{T-1} \mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t)$ the penalty on $\boldsymbol{S}$. The following propositions extend the result in Bemporad et al. (2018) to continuous jump models.

The first proposition states that, if $(\boldsymbol{Y}, \boldsymbol{S})$ follows a GHMM and the loss functions in (16) correspond to the negative log-likelihood, then maximizing a lower bound of the joint log-likelihood function is equivalent to minimizing the objective function J in the continuous jump model.

Proposition 1. *Suppose assumptions 1 through 3 above are satisfied, and define the terms in the loss function J as*
$$\begin{aligned} l(\boldsymbol{y}, \boldsymbol{\theta}) &:= -\log p(\boldsymbol{y}|\boldsymbol{\theta}), \\ \mathcal{L}^{init}(\boldsymbol{s}_0) &:= -\log \pi(\boldsymbol{s}_0), \\ \mathcal{L}^{jump}(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t) &:= -\log K(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t). \end{aligned} \tag{17}$$
Then maximizing a lower bound of the joint likelihood $p(\boldsymbol{Y}, \boldsymbol{S}|\Theta)$ is equivalent to minimizing the objective function J with respect to Θ and $\boldsymbol{S}$.

Conversely, given an objective function J, there is a statistical interpretation of it. In particular, the following proposition states that if certain assumptions are satisfied, including the use of probability density functions $p(\boldsymbol{y}|\boldsymbol{\theta})$ and $p(\boldsymbol{S})$, then minimizing the objective function J is equivalent to maximizing a lower bound of the joint likelihood.

Proposition 2. *Define the probability density functions*
$$p(\boldsymbol{y}|\boldsymbol{\theta}) := \frac{e^{-l(\boldsymbol{y}, \boldsymbol{\theta})}}{\nu}\,, \tag{18a}$$
$$p(\boldsymbol{S}) := \frac{e^{-\mathcal{L}(\boldsymbol{S})}}{\eta}\,, \tag{18b}$$

where

$$\nu := \int e^{-l(\boldsymbol{y},\boldsymbol{\theta})}\,d\boldsymbol{y} \text{ is assumed to be independent of } \boldsymbol{\theta}\text{, and}$$

$$\begin{aligned}\eta &:= \int_{\Delta_{K-1}^T} e^{-\mathcal{L}(\boldsymbol{S})}\,d\boldsymbol{S} \\ &= \int \cdots \int_{\Delta_{K-1}^T} e^{-\mathcal{L}^{init}(\boldsymbol{s}_0)} \prod_{t=1}^{T-1} e^{-\mathcal{L}^{jump}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t)}\,d\boldsymbol{s}_{T-1}\cdots d\boldsymbol{s}_0\,.\end{aligned} \tag{19}$$

Suppose assumptions 2 and 3 above are satisfied with (18a) being the parametric family, and (18b) is the density function for $\boldsymbol{S}$. Then minimizing the objective function J with respect to Θ and $\boldsymbol{S}$ is equivalent to maximizing a lower bound of the joint likelihood $p(\boldsymbol{Y},\boldsymbol{S}|\Theta)$.

We remark that the assumption that ν does not depend on $\boldsymbol{\theta}$ in the proposition above is satisfied if, for example, $l(\boldsymbol{y},\boldsymbol{\theta}) = g(\boldsymbol{y}-\boldsymbol{\theta})$.

3.4 Mode Loss

Inspired by the probabilistic interpretation of the continuous jump model in the previous section, we introduce another penalty on each individual hidden vector $\boldsymbol{s}_{t-1}$, called *mode loss*. Specifically, given a jump penalty $\mathcal{L}^{\text{jump}}$ in the loss function J, according to Equation (18b) in Proposition 2, the vector sequence $\boldsymbol{S}$ can be associated with a likelihood function

$$p(\boldsymbol{S}) \propto e^{-\mathcal{L}(\boldsymbol{S})} = e^{-\mathcal{L}^{\text{init}}(\boldsymbol{s}_0)} \prod_{t=1}^{T-1} e^{-\mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t)}. \tag{20}$$

Then, for $\boldsymbol{S}$ to follow a Markov process, the transition kernel $K(\boldsymbol{s}_{t-1},\boldsymbol{s}_t)$ needs to be proportional to $e^{-\mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t)}$, with a normalization constant given by

$$C(\boldsymbol{s}_{t-1}) = \int_{\Delta_{K-1}} e^{-\mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t)}\mathrm{d}\boldsymbol{s}_t\,. \tag{21}$$

Therefore, the transition kernel can be expressed as

$$K(\boldsymbol{s}_{t-1},\boldsymbol{s}_t) = \frac{1}{C(\boldsymbol{s}_{t-1})} e^{-\mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t)}\,. \tag{22}$$

Conversely, Proposition 1 suggests that the appropriate jump penalty for this transition kernel is

$$\widetilde{\mathcal{L}}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t) := -\log K(\boldsymbol{s}_{t-1},\boldsymbol{s}_t) \tag{23}$$

$$= \mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t) + \log C(\boldsymbol{s}_{t-1})\,. \tag{24}$$

Following Bemporad et al. (2018), we refer to the additional penalty term as the *mode loss*, defined as

$$\mathcal{L}^{\text{mode}}(\boldsymbol{s}_{t-1}) := \log C(\boldsymbol{s}_{t-1}) = \log \int_{\Delta_{K-1}} e^{-\mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t)}\mathrm{d}\boldsymbol{s}_t\,. \tag{25}$$

The full optimization problem with the mode loss becomes

$$\operatorname*{arg\,min}_{\Theta,\boldsymbol{S}} \sum_{t=0}^{T-1} L(\boldsymbol{y}_t,\Theta,\boldsymbol{s}_t) + \sum_{t=1}^{T-1} \mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t) + \sum_{t=1}^{T-1} \mathcal{L}^{\text{mode}}(\boldsymbol{s}_{t-1})\,. \tag{26}$$

When the mode loss $\mathcal{L}^{\text{mode}}(\boldsymbol{s}_{t-1})$ is large, then the model penalizes staying in the same state vector at time t. In contrast, with no mode loss this penalty is not present. With the jump penalty $\mathcal{L}^{\text{jump}}(\boldsymbol{s}_{t-1},\boldsymbol{s}_t) \propto \|\boldsymbol{s}_{t-1}-\boldsymbol{s}_t\|_1^2$, we observe that the vectors at the corners of the simplex (e.g. $(1,0,\ldots,0)$) have a small mode loss. They represent probability vectors with a high confidence of the current regime. On the other hand, the vectors around the center of the simplex (e.g. $(1/K, \ldots, 1/K)$), which represent probability vectors with a low confidence, have a large mode loss. Hence, the mode loss promotes sparsity in the state vectors, pushing the vectors to those with only a few non-zero elements. In empirical experiments, we observe that the mode loss penalty improves predictive performance, particularly during periods of state transitions. Therefore, if sparsity in the estimated probability vector is desirable for a particular application, we advise to include the mode loss in the objective function. However, a QP solver cannot handle the form of (25), and we must resort to the DP approximation approach. As we discretize the simplex, we can approximate the integration in (25) by

$$\mathcal{L}^{\text{mode}}(\boldsymbol{c}_i) \approx \log\left(\frac{1}{N}\sum_j e^{-\mathcal{L}^{\text{jump}}(\boldsymbol{c}_i,\boldsymbol{c}_j)}\right) = \log\left(\frac{1}{N}\sum_j e^{-\Lambda_{i,j}}\right) \tag{27}$$

$$= \operatorname{logsumexp}(-\Lambda_{i,\cdot}) + \text{const}\,, \tag{28}$$

where $\Lambda \in \mathbb{R}^{N\times N}$ is the penalty matrix in Algorithm 5, and $\operatorname{logsumexp}(\boldsymbol{x}) := \log\left(\sum_j e^{x_j}\right)$. It is straightforward to incorporate these changes into Algorithm 5. In particular, we modify the matrix Λ by first computing $\mathcal{L}^{\text{mode}}(\boldsymbol{c}_i)$, which is the logsumexp of the negative of the i-th row of Λ (neglecting the constant), and then adding this value to the i-th row to obtain the modified jump penalty matrix $\widetilde{\Lambda}$.

4 Simulation Study

We conduct a simulation study to compare the performance of the HMM estimated using the Baum-Welch algorithm, the discrete jump model, and the continuous jump model with and without the mode loss. In our unsupervised problem setting, we simulate data from well- and misspecified two- and three-state models where the true hidden state sequences and model parameters are known, thereby allowing us to evaluate the accuracy of different approaches in identifying the state sequences and in recovering the model parameters.

When evaluating classification accuracy in the context of financial time series, it is important to consider the high imbalance among regimes, where the majority of periods are in the bull (normal) regime rather than the bear regime. As a result, relying solely on overall accuracy frequently leads to inflated scores that do not accurately reflect a model's performance. To address this, we employ three additional metrics: accuracy per class, balanced accuracy (BAC), and the area under the Receiver Operating Characteristic curve (ROC-AUC). Accuracy per class measures the true positive rate for each individual class, while BAC reflects the average accuracy across all the classes, giving equal weight to each class irrespective of its proportion in the dataset (Brodersen et al., 2010). Breaking down the accuracy by class allows for a more comprehensive evaluation of the model's performance across different regimes. While the first two metrics, accuracy per class and BAC, are derived from the estimated label sequence, ROC-AUC is based on the estimated probability and assesses the model's ranking ability. Specifically, ROC-AUC represents the probability that a classifier assigns a higher estimated probability to a randomly selected positive instance compared to a randomly selected negative instance (Bradley, 1997). Unlike BAC, ROC-AUC can only be computed for the sequences where all of the states are present.

In the case of the discrete jump model, which performs hard clustering, we treat the estimated labels as probabilities of either zero or one when computing ROC-AUC. Notably, with two regimes, ROC-AUC is then equal to BAC. For the continuous jump model, which estimates probability vectors, we assign the predicted regime to be the one with the highest estimated probability. For HMMs, whether using true or estimated parameters, the predicted label sequence is obtained through the Viterbi algorithm, while the estimated probability refers to the smoothing posterior probability calculated using the forward-backward algorithm (Rabiner, 1989). In clustering-like problems, the cluster structure remains invariant to permutations of label symbols. Therefore, we select the permutation that yields the highest overall accuracy.

4.1 Two-State HMM

Following Zheng et al. (2021), we simulate data from the two-state Gaussian HMM

$$y_t | s_t \sim \mathcal{N}\left(\mu_{s_t}, \sigma^2_{s_t}\right), \tag{29}$$

where the state sequence $\{s_t\}$ follows a first-order Markov chain and the state-conditional parameters and transition probability matrix are given by

$$\begin{aligned}
&\mu_1 = .0123, \quad \mu_2 = -.0157, \\
&\sigma_1 = .0347, \quad \sigma_2 = .0778, \\
&\boldsymbol{P} = \begin{bmatrix} .9629 & .0371 \\ .2102 & .7899 \end{bmatrix}.
\end{aligned} \tag{30}$$

In this two-state HMM, the first state can be interpreted as a bullish market regime, while the second state represents a bearish market regime. We note that Hardy (2001) estimated these parameters from the monthly returns of an equity market index. We choose the starting distribution of s_0 as the stationary distribution of the Markov chain.

To simulate observation sequences with different state persistence, we construct daily and weekly versions of the model above. Assuming a geometric Brownian motion for the return process and a continuous-time Markov chain for the hidden state process, the following relationship between the parameters for two time scales t_1 and t_2 holds

$$\begin{aligned}
&\mu_s(t_1)/t_1 = \mu_s(t_2)/t_2, \quad \sigma_s^2(t_1)/t_1 = \sigma_s^2(t_2)/t_2, \\
&\boldsymbol{P}(t_1)^{1/t_1} = \boldsymbol{P}(t_2)^{1/t_2},
\end{aligned} \tag{31}$$

where the last equation is expressed in terms of matrix exponentials. In our empirical work, we use $t = 1, 5, 20$ for the daily, weekly and monthly time scales, respectively. We remark that simulated daily data exhibits the highest level of persistence but also has the lowest signal-to-noise ratio. Here, the signal-to-noise ratio measures the separability among clusters, and is calculated as the ratio of the distance between two cluster centers to the volatility (Balakrishnan et al., 2017).

4.1.1 Selecting λ

Figure 1 displays the average BAC of the three jump models as a function of the penalty parameter for 1024 simulated sequences each of length 1000 using the daily parameters in (30). Consistent with previous findings (Nystrup et al., 2020a,b), the discrete model achieves the highest BAC at $\lambda = 10^2$. Interestingly, the continuous model, although it can be reduced to a discrete model with the same λ, achieves the best performance at $\lambda = 10^3$ and demonstrates better robustness to the hyperparameter. The BAC plots for sequences with different lengths exhibit similar patterns, except that the overall level of BAC is higher for longer sequences. While we omit the details, we find that the discrete and continuous models exhibit optimal λ values of around 50 and 10^2, respectively, at the weekly scale. At the monthly scale, both models perform best with a smaller value of λ of around one. These findings imply that in less persistent data, optimal performance is attained by imposing a smaller jump penalty.

4.1.2 Daily Scale

Table 1 presents the mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 250, 500, 1000, 2000$ from the daily two-state HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss. The bold values represent the best parameter estimates and highest accuracy scores. For the daily scale, the sequence lengths we consider correspond approximately to 1, 2, 4, and 8 years, respectively. It is worth noting that, similar to Nystrup et al. (2020b), we retain all the sequences without filtering out those that only contain a single regime, aiming to better emulate the real-world scenarios.

We estimate the HMMs with Baum-Welch algorithm implemented in the Python package `hmmlearn`. Inline with common practise to ensure numerical stability, for the transition probabilities we incorporate a Dirichlet prior distribution of order K where all parameters are set to $1+10^{-8}$. Additionally, we initialize the estimation by the K-means++ algorithm, similar to the initialization approach we use for the jump models. In the continuous jump model when fitting the hidden vector sequence through the coordinate descent algorithm, we employ the approximate DP approach and discretize the probability simplex uniformly with a grid size of 0.01 for all two-state models.

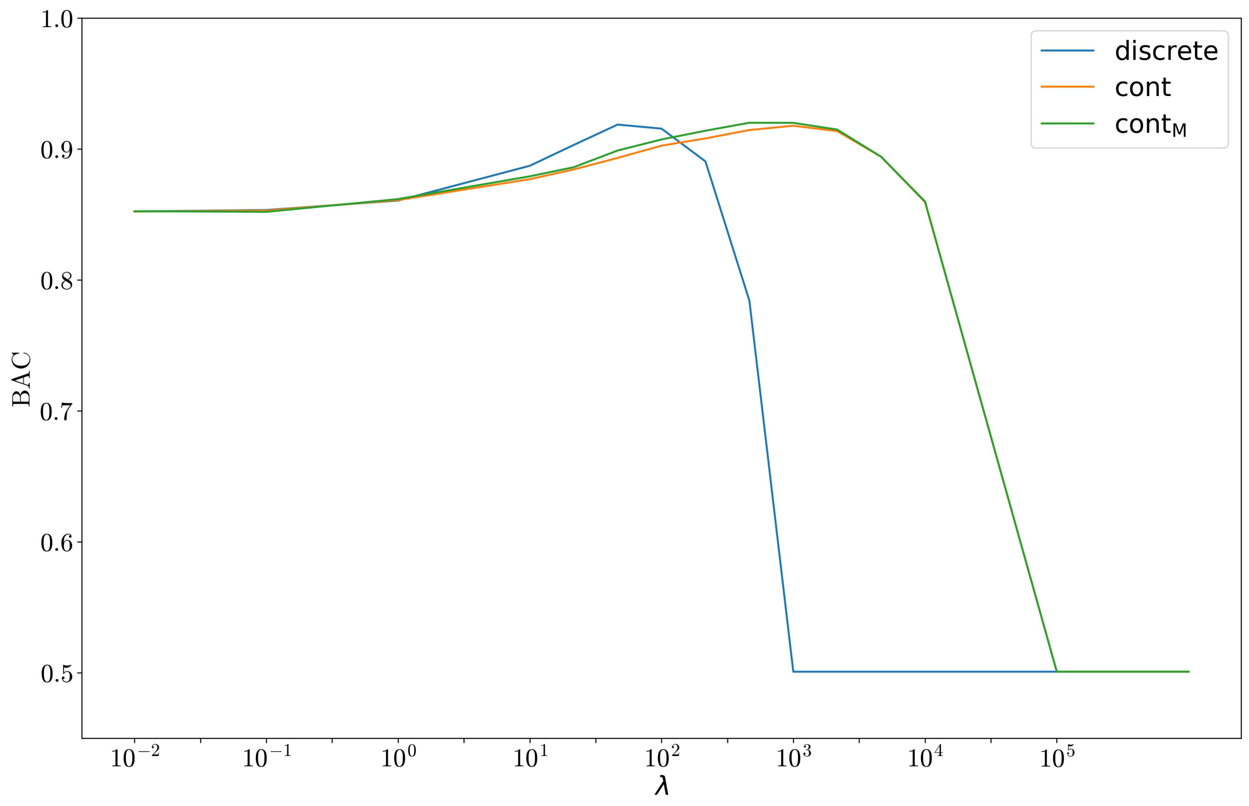

Figure 1: Balanced accuracy of the jump models as a function of the penalty parameter for 1024 simulated sequences each of length 1000. discrete, cont, and cont_M denote the discrete and continuous jump models without and with mode loss, respectively.

When examining the accuracy of parameter estimation, a notable distinction between HMM and jump models is in the estimation of transition probabilities. Specifically, HMM significantly overestimates the transition probabilities, often by tens of multiples compared to their true values, especially for shorter time series. In addition, even as the length of the time series increases, the convergence of the estimated HMM to the true value is very slow. In contrast, jump models demonstrate the ability to capture the true persistence in the state sequence, even for shorter time series, thereby significantly outperforming the HMMs in terms of BAC and ROC-AUC. The performance of the HMMs are on par with the jump models only when the time series are longer, e.g. more than 1000 observations. Notably, the considerably larger standard deviation of the estimated transition probabilities of the HMMs highlights a lack of robustness, despite the estimated models being well-specified. These findings are consistent with previous studies (Nystrup et al., 2020a,b).

While the MLE of HMMs is consistent, Bickel et al. (1998) demonstrate that the convergence rate to the true parameters is $O\left(1/\sqrt{T}\right)$, where T is the length of the time series. Based on our empirical results, this implies that achieving accurate parameter estimates would require an observation sequence of several multiples of 2000. However, in financial applications with daily financial data this would be challenging practically (Nystrup et al., 2017, 2020a). Recent theoretical work on the statistical properties of Baum-Welch algorithms provides insights into the slow convergence rate observed in practice (Yang et al., 2017; Balakrishnan et al., 2017). This slow rate can be attributed to the stylized facts observed in financial time series, which include a low signal-to-noise ratio due to significant overlap between states, high persistence, and large imbalances of observations from different regimes.

Comparing the discrete and continuous jump models, we observe that the continuous model achieves a similar BAC to the discrete model, while significantly improving the ROC-AUC. This indicates that, in addition to providing commensurate labeling results, the continuous model also offers reliable probability estimates for each regime. This gives us confidence in applying the estimated probabilities to downstream tasks such as portfolio optimization and risk management applications.

When comparing the two versions of the continuous model, we notice that the inclusion of the mode loss slightly improves the model's performance in small sample sizes. This supports the notion that the mode loss promotes sparsity and persistence in the estimated probability vector. However, with sufficient data, the impact of the mode loss becomes less apparent.

4.1.3 Weekly Scale

Table 2 presents the mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 50, 100, 250, 500$ from the weekly two-state HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss. For the weekly scale, the sequence lengths we consider correspond approximately to 1, 2, 5, and 10 years, respectively.

By going from the daily to the weekly scale, the number of time series observations decrease by a factor of five, making it significantly more challenging to fit an HMM. Despite having the equivalent of about ten years of weekly observations when $T = 500$, the estimated transition probabilities in the HMMs deviates significantly from the true ones, resulting in a lack of persistence of the state sequence. In contrast, the jump models capture the true persistence of the underlying state sequence. The performance difference between the discrete and continuous models is similar to the daily results above. Specifically, while the discrete and continuous jump models obtain comparable BACs, the continuous jump models exhibit better ROC-AUCs. Notably, due to the reduced level of persistence, all models exhibit lower accuracy in detecting the bear market regime compared to their daily counterparts.

4.1.4 Monthly Scale

Table 3 presents the mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 60, 120, 250, 500$ from the monthly two-state HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss. For the monthly scale, the sequence lengths we consider correspond approximately to 2, 10, 20 and 40 years, respectively.

Consistent with the empirical results at the daily and weekly scales, the jump models achieve higher BAC and more accurate parameter estimates when the number of observations is limited. Due to the relatively low persistence of monthly data, the estimates of the transition probabilities for the HMMs with a sequence

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC	ROC-AUC
250										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9872 (0.0893)	0.8481 (0.2895)	0.9177 (0.1456)	0.9905 (0.0577)
HMM	0.0005 (0.0043)	-0.0013 (0.0074)	**0.0078** (0.0018)	0.0104 (0.0061)	0.1960 (0.2731)	0.4525 (0.3961)	0.8410 (0.2182)	0.7642 (0.3132)	0.8026 (0.2007)	0.8477 (0.2397)
discrete	0.0006 (0.0008)	**-0.0003** (0.0031)	0.0080 (0.0015)	**0.0140** (0.0048)	**0.0031** (0.0051)	0.0182 (0.0266)	**0.9217** (0.1514)	0.8123 (0.2758)	**0.8670** (0.1537)	0.8778 (0.1533)
cont	0.0006 (0.0008)	0.0001 (0.0022)	0.0082 (0.0016)	0.0118 (0.0044)	0.0034 (0.0048)	**0.0125** (0.0233)	0.8782 (0.1667)	0.8175 (0.2955)	0.8478 (0.1704)	**0.9391** (0.1481)
cont_M	**0.0006** (0.0008)	0.0001 (0.0024)	0.0081 (0.0016)	0.0121 (0.0046)	0.0042 (0.0049)	0.0169 (0.0273)	0.8670 (0.1616)	**0.8434** (0.2558)	0.8552 (0.1593)	0.9380 (0.1500)
500										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9955 (0.0154)	0.8685 (0.2495)	0.9320 (0.1238)	0.9915 (0.0591)
HMM	0.0006 (0.0021)	**-0.0008** (0.0065)	**0.0077** (0.0009)	0.0130 (0.0057)	0.1159 (0.2046)	0.2931 (0.3711)	0.9102 (0.1622)	0.8228 (0.2827)	0.8665 (0.1731)	0.9229 (0.1926)
discrete	0.0006 (0.0005)	-0.0005 (0.0028)	0.0079 (0.0005)	**0.0148** (0.0043)	**0.0022** (0.0021)	0.0163 (0.0193)	**0.9457** (0.1020)	0.8294 (0.2407)	**0.8876** (0.1355)	0.9023 (0.1284)
cont	**0.0006** (0.0005)	-0.0002 (0.0022)	0.0080 (0.0005)	0.0136 (0.0044)	0.0031 (0.0022)	**0.0147** (0.0157)	0.9017 (0.1355)	**0.8598** (0.2146)	0.8808 (0.1388)	**0.9575** (0.1266)
cont_M	0.0006 (0.0005)	-0.0002 (0.0023)	0.0080 (0.0005)	0.0137 (0.0044)	0.0027 (0.0021)	0.0153 (0.0192)	0.9178 (0.1228)	0.8529 (0.2205)	0.8854 (0.1371)	0.9531 (0.1234)
1000										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9957 (0.0123)	0.9049 (0.1819)	0.9503 (0.0905)	0.9956 (0.0321)
HMM	**0.0006** (0.0006)	-0.0004 (0.0039)	**0.0077** (0.0006)	0.0155 (0.0043)	0.0491 (0.1486)	0.1341 (0.2739)	0.9656 (0.1060)	**0.8905** (0.2006)	**0.9280** (0.1249)	0.9616 (0.1414)
discrete	0.0006 (0.0003)	**-0.0007** (0.0023)	0.0079 (0.0003)	**0.0163** (0.0031)	0.0019 (0.0013)	0.0154 (0.0162)	**0.9744** (0.0606)	0.8566 (0.1840)	0.9155 (0.1003)	0.9212 (0.0970)
cont	0.0006 (0.0003)	-0.0005 (0.0019)	0.0079 (0.0003)	0.0156 (0.0033)	0.0023 (0.0016)	**0.0137** (0.0108)	0.9539 (0.0969)	0.8816 (0.1662)	0.9177 (0.1059)	**0.9710** (0.1011)
cont_M	0.0006 (0.0003)	-0.0006 (0.0019)	0.0079 (0.0003)	0.0157 (0.0033)	**0.0021** (0.0014)	0.0141 (0.0120)	0.9636 (0.0783)	0.8763 (0.1719)	0.9199 (0.1037)	0.9674 (0.0938)
2000										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9965 (0.0061)	0.9212 (0.1230)	0.9588 (0.0612)	0.9975 (0.0157)
HMM	**0.0006** (0.0003)	-0.0010 (0.0037)	**0.0078** (0.0004)	0.0166 (0.0028)	0.0266 (0.1040)	0.0738 (0.1971)	0.9819 (0.0733)	**0.8946** (0.1773)	**0.9382** (0.1096)	0.9693 (0.1185)
discrete	0.0006 (0.0002)	**-0.0008** (0.0016)	0.0079 (0.0002)	**0.0171** (0.0018)	0.0017 (0.0009)	**0.0136** (0.0111)	**0.9899** (0.0190)	0.8717 (0.1388)	0.9308 (0.0707)	0.9314 (0.0702)
cont	0.0006 (0.0002)	-0.0007 (0.0014)	0.0079 (0.0002)	0.0168 (0.0020)	**0.0021** (0.0015)	0.0140 (0.0091)	0.9795 (0.0607)	0.8926 (0.1143)	0.9360 (0.0688)	**0.9768** (0.0628)
cont_M	0.0006 (0.0002)	-0.0007 (0.0014)	0.0079 (0.0002)	0.0168 (0.0019)	0.0020 (0.0011)	0.0141 (0.0095)	0.9846 (0.0391)	0.8885 (0.1219)	0.9366 (0.0685)	0.9712 (0.0641)

Table 1: Mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 250, 500, 1000, 2000$ from the daily two-state HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss (discrete, cont, and cont_M). The bold values represent the best parameter estimates and highest accuracy scores.

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC	ROC-AUC
50										
true	0.0031 (0.0000)	-0.0039 (0.0000)	0.0174 (0.0000)	0.0389 (0.0000)	0.0103 (0.0000)	0.0582 (0.0000)	0.9727 (0.1179)	0.6019 (0.4176)	0.7873 (0.1932)	0.9421 (0.1387)
HMM	**0.0032** (0.0155)	**-0.0029** (0.0260)	0.0159 (0.0051)	0.0164 (0.0127)	0.2775 (0.2908)	0.6081 (0.3652)	0.7533 (0.2248)	0.5371 (0.3276)	0.6452 (0.1688)	0.6592 (0.2504)
discrete	0.0029 (0.0046)	0.0001 (0.0103)	**0.0180** (0.0045)	**0.0250** (0.0110)	**0.0140** (0.0207)	0.0436 (0.0531)	**0.8049** (0.2002)	0.6593 (0.3702)	**0.7321** (0.1917)	0.7377 (0.1992)
cont	0.0028 (0.0040)	0.0008 (0.0074)	0.0185 (0.0047)	0.0232 (0.0093)	0.0182 (0.0192)	0.0408 (0.0582)	0.7431 (0.1708)	**0.7173** (0.3397)	0.7302 (0.1907)	**0.8087** (0.2358)
cont_{M}	0.0029 (0.0043)	0.0008 (0.0082)	0.0183 (0.0048)	0.0235 (0.0098)	0.0201 (0.0198)	**0.0471** (0.0587)	0.7498 (0.1674)	0.7095 (0.3302)	0.7296 (0.1923)	0.7981 (0.2210)
100										
true	0.0031 (0.0000)	-0.0039 (0.0000)	0.0174 (0.0000)	0.0389 (0.0000)	0.0103 (0.0000)	0.0582 (0.0000)	0.9822 (0.0639)	0.6273 (0.3816)	0.8048 (0.1842)	0.9512 (0.1235)
HMM	0.0028 (0.0094)	**-0.0031** (0.0243)	**0.0167** (0.0044)	0.0220 (0.0144)	0.2085 (0.2717)	0.5116 (0.3813)	0.8178 (0.2089)	0.5616 (0.3352)	0.6897 (0.1872)	0.7220 (0.2722)
discrete	**0.0030** (0.0027)	-0.0009 (0.0097)	0.0180 (0.0029)	**0.0282** (0.0112)	**0.0097** (0.0086)	**0.0424** (0.0459)	**0.8527** (0.1545)	0.6585 (0.3392)	0.7556 (0.1877)	0.7743 (0.1853)
cont	0.0029 (0.0027)	-0.0002 (0.0077)	0.0181 (0.0029)	0.0269 (0.0102)	0.0149 (0.0106)	0.0408 (0.0311)	0.8052 (0.1581)	**0.7202** (0.3082)	0.7627 (0.1870)	**0.8394** (0.1974)
cont_{M}	0.0029 (0.0027)	-0.0001 (0.0079)	0.0182 (0.0029)	0.0269 (0.0103)	0.0135 (0.0102)	0.0406 (0.0335)	0.8175 (0.1538)	0.7082 (0.3176)	**0.7629** (0.1886)	0.8263 (0.1974)
250										
true	0.0031 (0.0000)	-0.0039 (0.0000)	0.0174 (0.0000)	0.0389 (0.0000)	0.0103 (0.0000)	0.0582 (0.0000)	0.9857 (0.0286)	0.6924 (0.2893)	0.8390 (0.1415)	0.9668 (0.0810)
HMM	0.0031 (0.0024)	**-0.0035** (0.0176)	**0.0175** (0.0021)	0.0310 (0.0129)	0.0964 (0.2054)	0.2943 (0.3431)	**0.9268** (0.1371)	0.6379 (0.3107)	0.7824 (0.1809)	0.8403 (0.2386)
discrete	0.0032 (0.0017)	-0.0033 (0.0083)	0.0178 (0.0016)	**0.0336** (0.0088)	0.0125 (0.0086)	**0.0532** (0.0289)	0.8987 (0.1238)	0.7144 (0.2243)	0.8065 (0.1317)	0.8143 (0.1281)
cont	**0.0031** (0.0016)	-0.0026 (0.0073)	0.0181 (0.0016)	0.0327 (0.0086)	**0.0119** (0.0083)	0.0457 (0.0235)	0.8886 (0.1330)	0.7259 (0.2281)	0.8073 (0.1377)	**0.8627** (0.1452)
cont_{M}	0.0032 (0.0017)	-0.0028 (0.0074)	0.0180 (0.0016)	0.0330 (0.0087)	0.0130 (0.0091)	0.0503 (0.0255)	0.8900 (0.1308)	**0.7261** (0.2200)	**0.8080** (0.1316)	0.8445 (0.1321)
500										
true	0.0031 (0.0000)	-0.0039 (0.0000)	0.0174 (0.0000)	0.0389 (0.0000)	0.0103 (0.0000)	0.0582 (0.0000)	0.9879 (0.0165)	0.7549 (0.2009)	0.8714 (0.0986)	0.9777 (0.0434)
HMM	**0.0031** (0.0013)	-0.0045 (0.0127)	**0.0174** (0.0012)	0.0360 (0.0090)	0.0416 (0.1287)	0.1640 (0.2468)	**0.9665** (0.0888)	0.7150 (0.2421)	**0.8408** (0.1351)	**0.9209** (0.1638)
discrete	0.0032 (0.0011)	**-0.0042** (0.0064)	0.0179 (0.0010)	**0.0364** (0.0060)	**0.0096** (0.0056)	**0.0529** (0.0220)	0.9419 (0.0785)	0.7222 (0.1618)	0.8321 (0.0912)	0.8324 (0.0910)
cont	0.0031 (0.0011)	-0.0034 (0.0057)	0.0181 (0.0010)	0.0354 (0.0058)	0.0090 (0.0054)	0.0457 (0.0175)	0.9357 (0.0846)	**0.7394** (0.1594)	0.8375 (0.0935)	0.8899 (0.0969)
cont_{M}	0.0030 (0.0010)	-0.0036 (0.0058)	0.0182 (0.0010)	0.0357 (0.0057)	0.0082 (0.0044)	0.0455 (0.0187)	0.9428 (0.0742)	0.7289 (0.1664)	0.8359 (0.0944)	0.8739 (0.0984)

Table 2: Mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 50, 100, 250, 500$ from the weekly two-state HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss (discrete, cont, and cont_{M}). The bold values represent the best parameter estimates and highest accuracy scores.

length of 500 are the best. In comparison to the results in Nystrup et al. (2020b), our jump models do not achieve a higher BAC than the true model. This suggests that the outperformance reported in Nystrup et al. (2020b) is likely due to that some of their features are forward-looking. We emphasize that as all of our features are backward-looking, our model can be applied in an online forcasting fashion while still achieving a BAC close to that of the true model. Additionally, the jump models demonstrate higher accuracy for the bear regime than the HMMs.

It is worth noting that while the ROC-AUCs of the HMMs are significantly higher than those of the jump models once the number of observations exceed 250, there is an important difference. The labels and probabilities from HMMs are calculated using two different algorithms, the Viterbi algorithm and the forward-backward algorithm. As a result, they may not be consistent with each other, meaning that the regime with the maximum estimated probability from the forward-backward algorithm may not align with the label from the Viterbi algorithm. In contrast, our continuous jump model ensures consistency between the labels and estimated probabilities, as we assign the label corresponding to the regime with the highest estimated probability. This presents a notable benefit of our approach.

4.2 Three-State HMM

Several studies suggest HMMs with more than two states can provide a better fit to financial return series, based on information criteria such as the Bayesian Information Criterion (BIC) (Bulla, 2011; Nystrup et al., 2015; Dias et al., 2015; Cortese et al., 2023a,b). As the number of states increases, the estimation of HMMs becomes considerably more challenging due to the quadratic growth of parameters in the number of states.

To assess the performance as the number of states increases, we follow Zheng et al. (2021) by augmenting the two-state model with a third intermediate state. This augmentation results in a model with the following parameters

$$\begin{aligned} &\mu_1 = .0123, \quad \mu_2 = .0000, \quad \mu_3 = -.0157, \\ &\sigma_1 = .0347, \quad \sigma_2 = .0500, \quad \sigma_3 = .0778, \\ &\boldsymbol{P} = \begin{bmatrix} .9629 & .0185 & .0186 \\ .0618 & .8764 & .0618 \\ .1051 & .1050 & .7899 \end{bmatrix}. \end{aligned} \tag{32}$$

The parameters can be transformed to different scales using using the formulas in equation (31).

As the ROC curve is typically defined for binary classification problems, in the case of the three-state model, we compute the ROC-AUC as the unweighted average area under the ROC curves for all possible pairwise comparisons among the states (Hand and Till, 2001). This allows us to assess the model's ranking ability in a multi-state setting and provide a comprehensive evaluation of its performance across different class combinations.

Table 4 presents the mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 500, 1000, 2000$ from the daily three-state HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss. We do not show results for a sequence length of 250 as it is insufficient for accurately estimating a persistent model. We only present results for the daily scale, as the conclusions are consistent across weekly and monthly parameters. For the continuous jump model, we discretize the probability simplex using a uniform grid with a size of 0.05.

The addition of a third state significantly increases the difficulty of estimating the HMMs. Even for the longest time series of 2000 observations, it is not possible to capture the true persistence accurately. In contrast, we observe that the jump models obtain the best parameter estimates for almost all of the twelve model parameters. When comparing the discrete and continuous jump models, we note that the continuous model achieves an improvement of 2% and 4% in BAC and ROC-AUC, respectively. This validates the effectiveness of the continuous model, as it allows the representation of hidden states as continuous probabilities, contributing to increased accuracy.

The BAC of all models increases slowly with the length of the time series, underscoring the difficulty of precisely fitting models with more than two states. However, compared to the results in Nystrup et al. (2020b), our jump models achieve higher accuracy even when the feature set is limited to solely backward-looking features.

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC	ROC-AUC
60										
true	0.0123 (0.0000)	-0.0157 (0.0000)	0.0347 (0.0000)	0.0778 (0.0000)	0.0371 (0.0000)	0.2101 (0.0000)	0.9821 (0.0410)	0.4389 (0.3465)	0.7105 (0.1669)	0.8820 (0.1674)
HMM	**0.0124** (0.0109)	**-0.0172** (0.0613)	0.0332 (0.0082)	0.0397 (0.0285)	0.2287 (0.2799)	0.6212 (0.3520)	**0.8181** (0.1815)	0.4459 (0.3115)	0.6320 (0.1654)	0.6415 (0.2450)
discrete	0.0125 (0.0082)	-0.0031 (0.0225)	**0.0343** (0.0059)	0.0558 (0.0201)	0.0678 (0.0467)	**0.1325** (0.0644)	0.7517 (0.1478)	**0.6498** (0.2886)	**0.7007** (0.1605)	**0.7075** (0.1599)
cont	0.0125 (0.0083)	-0.0033 (0.0223)	0.0342 (0.0059)	**0.0558** (0.0198)	**0.0674** (0.0471)	0.1316 (0.0631)	0.7524 (0.1475)	0.6477 (0.2916)	0.7000 (0.1615)	0.7068 (0.1609)
cont$_{\mathrm{M}}$	0.0125 (0.0083)	-0.0032 (0.0224)	0.0342 (0.0058)	0.0557 (0.0199)	0.0679 (0.0473)	0.1322 (0.0646)	0.7512 (0.1463)	0.6450 (0.2903)	0.6981 (0.1611)	0.7048 (0.1605)
120										
true	0.0123 (0.0000)	-0.0157 (0.0000)	0.0347 (0.0000)	0.0778 (0.0000)	0.0371 (0.0000)	0.2101 (0.0000)	0.9829 (0.0267)	0.4696 (0.2650)	0.7263 (0.1282)	0.8969 (0.1002)
HMM	**0.0120** (0.0069)	-0.0260 (0.0548)	0.0345 (0.0060)	0.0560 (0.0293)	0.1412 (0.2159)	0.4881 (0.3414)	**0.8913** (0.1526)	0.4577 (0.2721)	0.6745 (0.1453)	**0.7267** (0.2230)
discrete	0.0131 (0.0063)	-0.0079 (0.0201)	**0.0347** (0.0045)	**0.0630** (0.0172)	0.0580 (0.0407)	**0.1402** (0.0523)	0.7969 (0.1353)	0.6252 (0.2213)	0.7111 (0.1199)	0.7117 (0.1198)
cont	0.0132 (0.0062)	-0.0080 (0.0201)	0.0346 (0.0045)	0.0630 (0.0173)	0.0584 (0.0410)	0.1399 (0.0517)	0.7960 (0.1351)	0.6269 (0.2208)	0.7114 (0.1185)	0.7120 (0.1183)
cont$_{\mathrm{M}}$	0.0132 (0.0063)	**-0.0081** (0.0205)	0.0346 (0.0045)	0.0629 (0.0172)	**0.0578** (0.0405)	0.1389 (0.0511)	0.7967 (0.1357)	**0.6304** (0.2184)	**0.7135** (0.1166)	0.7141 (0.1164)
250										
true	0.0123 (0.0000)	-0.0157 (0.0000)	0.0347 (0.0000)	0.0778 (0.0000)	0.0371 (0.0000)	0.2101 (0.0000)	0.9836 (0.0186)	0.5105 (0.1842)	0.7470 (0.0894)	0.9120 (0.0583)
HMM	**0.0125** (0.0033)	-0.0258 (0.0365)	0.0343 (0.0032)	**0.0694** (0.0205)	0.0665 (0.1098)	0.3350 (0.2599)	**0.9539** (0.0886)	0.5080 (0.2182)	0.7310 (0.1084)	**0.8483** (0.1473)
discrete	0.0136 (0.0043)	**-0.0122** (0.0139)	0.0349 (0.0031)	0.0650 (0.0129)	**0.0520** (0.0312)	0.1570 (0.0433)	0.8368 (0.1041)	0.6391 (0.1515)	0.7380 (0.0789)	0.7380 (0.0789)
cont	0.0136 (0.0042)	-0.0121 (0.0138)	0.0349 (0.0031)	0.0650 (0.0130)	0.0523 (0.0313)	0.1574 (0.0437)	0.8360 (0.1044)	**0.6405** (0.1509)	**0.7382** (0.0795)	0.7382 (0.0794)
cont$_{\mathrm{M}}$	0.0136 (0.0044)	-0.0121 (0.0138)	**0.0349** (0.0031)	0.0650 (0.0130)	0.0525 (0.0315)	**0.1576** (0.0437)	0.8356 (0.1050)	0.6403 (0.1508)	0.7380 (0.0795)	0.7380 (0.0795)
500										
true	0.0123 (0.0000)	-0.0157 (0.0000)	0.0347 (0.0000)	0.0778 (0.0000)	0.0371 (0.0000)	0.2101 (0.0000)	0.9842 (0.0135)	0.5194 (0.1291)	0.7518 (0.0627)	0.9135 (0.0377)
HMM	**0.0123** (0.0022)	**-0.0151** (0.0254)	**0.0346** (0.0025)	**0.0739** (0.0154)	0.0554 (0.1050)	0.2785 (0.1928)	**0.9678** (0.0670)	0.5136 (0.1810)	0.7407 (0.0895)	**0.8810** (0.1108)
discrete	0.0138 (0.0029)	-0.0134 (0.0095)	0.0351 (0.0022)	0.0663 (0.0092)	0.0473 (0.0234)	**0.1668** (0.0356)	0.8597 (0.0710)	0.6301 (0.1047)	0.7449 (0.0523)	0.7449 (0.0523)
cont	0.0137 (0.0028)	-0.0134 (0.0096)	0.0351 (0.0022)	0.0663 (0.0091)	0.0471 (0.0231)	0.1666 (0.0352)	0.8600 (0.0702)	**0.6302** (0.1053)	**0.7451** (0.0517)	0.7451 (0.0517)
cont$_{\mathrm{M}}$	0.0138 (0.0028)	-0.0134 (0.0096)	0.0351 (0.0021)	0.0663 (0.0092)	**0.0471** (0.0233)	0.1664 (0.0356)	0.8600 (0.0707)	0.6294 (0.1057)	0.7447 (0.0524)	0.7447 (0.0524)

Table 3: Mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 60, 120, 250, 500$ from the monthly two-state HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss (discrete, cont, and cont$_{\mathrm{M}}$). The bold values represent the best parameter estimates and highest accuracy scores.

	μ_1	μ_2	μ_3	σ_1	σ_2	σ_3	γ_{12}	γ_{13}	γ_{21}
500									
true	0.0006 (0.0000)	0.0000 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0112 (0.0000)	0.0174 (0.0000)	0.0009 (0.0000)	0.0010 (0.0000)	0.0032 (0.0000)
HMM	0.0009 (0.0061)	0.0012 (0.0086)	-0.0017 (0.0073)	**0.0078** (0.0023)	0.0082 (0.0039)	0.0100 (0.0054)	0.2601 (0.3376)	0.2120 (0.3009)	0.3909 (0.3968)
discrete	0.0005 (0.0015)	-0.0000 (0.0029)	-0.0007 (0.0031)	0.0085 (0.0027)	**0.0115** (0.0033)	0.0145 (0.0042)	0.0023 (0.0093)	0.0020 (0.0069)	**0.0076** (0.0161)
cont	0.0005 (0.0011)	**0.0000** (0.0023)	**-0.0008** (0.0024)	0.0084 (0.0021)	0.0116 (0.0029)	0.0147 (0.0039)	0.0017 (0.0071)	0.0016 (0.0073)	0.0093 (0.0487)
cont_{M}	**0.0006** (0.0012)	-0.0000 (0.0023)	-0.0007 (0.0024)	0.0084 (0.0021)	0.0117 (0.0030)	**0.0148** (0.0038)	**0.0015** (0.0052)	**0.0015** (0.0058)	0.0084 (0.0458)
1000									
true	0.0006 (0.0000)	0.0000 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0112 (0.0000)	0.0174 (0.0000)	0.0009 (0.0000)	0.0010 (0.0000)	0.0032 (0.0000)
HMM	0.0007 (0.0033)	0.0008 (0.0073)	-0.0010 (0.0062)	0.0084 (0.0027)	0.0093 (0.0038)	0.0117 (0.0053)	0.2127 (0.3264)	0.1505 (0.2788)	0.3367 (0.4006)
discrete	**0.0006** (0.0014)	-0.0000 (0.0034)	**-0.0007** (0.0029)	0.0084 (0.0022)	0.0122 (0.0033)	**0.0155** (0.0038)	0.0015 (0.0052)	0.0017 (0.0054)	0.0065 (0.0132)
cont	0.0006 (0.0009)	0.0000 (0.0024)	-0.0006 (0.0023)	**0.0083** (0.0019)	0.0121 (0.0030)	0.0153 (0.0035)	**0.0011** (0.0034)	**0.0011** (0.0026)	0.0054 (0.0151)
cont_{M}	0.0006 (0.0008)	**0.0000** (0.0024)	-0.0006 (0.0023)	0.0084 (0.0019)	**0.0121** (0.0030)	0.0155 (0.0033)	0.0012 (0.0034)	0.0012 (0.0034)	**0.0052** (0.0139)
2000									
true	0.0006 (0.0000)	0.0000 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0112 (0.0000)	0.0174 (0.0000)	0.0009 (0.0000)	0.0010 (0.0000)	0.0032 (0.0000)
HMM	0.0005 (0.0022)	0.0008 (0.0072)	-0.0003 (0.0051)	0.0083 (0.0018)	**0.0105** (0.0041)	0.0125 (0.0048)	0.1329 (0.2858)	0.1120 (0.2575)	0.2290 (0.3704)
discrete	**0.0006** (0.0007)	-0.0001 (0.0034)	-0.0009 (0.0027)	0.0082 (0.0013)	0.0128 (0.0031)	**0.0165** (0.0026)	**0.0009** (0.0023)	**0.0011** (0.0020)	0.0056 (0.0099)
cont	0.0006 (0.0005)	-0.0002 (0.0021)	-0.0007 (0.0021)	**0.0081** (0.0010)	0.0124 (0.0026)	0.0163 (0.0025)	0.0008 (0.0019)	0.0008 (0.0015)	0.0055 (0.0158)
cont_{M}	0.0006 (0.0006)	**-0.0001** (0.0022)	**-0.0008** (0.0021)	0.0081 (0.0011)	0.0124 (0.0026)	0.0163 (0.0025)	0.0009 (0.0020)	0.0009 (0.0016)	**0.0048** (0.0088)

	γ_{23}	γ_{31}	γ_{32}	Accuracy 1	Accuracy 2	Accuracy 3	BAC	ROC-AUC
500								
true	0.0037 (0.0000)	0.0058 (0.0000)	0.0062 (0.0000)	0.9557 (0.1685)	0.7223 (0.3969)	0.7654 (0.3480)	0.8145 (0.1590)	0.9667 (0.0662)
HMM	0.2695 (0.3320)	0.3199 (0.3735)	0.2922 (0.3513)	0.6284 (0.3051)	0.4472 (0.3521)	0.5452 (0.3660)	0.5403 (0.1077)	0.6626 (0.1611)
discrete	0.0059 (0.0150)	0.0091 (0.0168)	0.0091 (0.0171)	0.8446 (0.2650)	0.6677 (0.3842)	0.6558 (0.3201)	0.7227 (0.1463)	0.7560 (0.1189)
cont	**0.0059** (0.0327)	**0.0071** (0.0130)	0.0084 (0.0297)	**0.8671** (0.2477)	0.7046 (0.3786)	0.6480 (0.3651)	0.7399 (0.1467)	**0.7968** (0.1274)
cont_{M}	0.0060 (0.0440)	0.0075 (0.0171)	**0.0072** (0.0182)	0.8645 (0.2560)	**0.7072** (0.3722)	**0.6564** (0.3577)	**0.7427** (0.1497)	0.7909 (0.1298)
1000								
true	0.0037 (0.0000)	0.0058 (0.0000)	0.0062 (0.0000)	0.9703 (0.1017)	0.7465 (0.3560)	0.8129 (0.2843)	0.8432 (0.1325)	0.9828 (0.0328)
HMM	0.2671 (0.3474)	0.2474 (0.3617)	0.2516 (0.3265)	0.7114 (0.3099)	0.4029 (0.3858)	0.5460 (0.4026)	0.5534 (0.1224)	0.6812 (0.1968)
discrete	0.0078 (0.0179)	0.0090 (0.0136)	0.0081 (0.0138)	0.8866 (0.2370)	0.6296 (0.3796)	0.6941 (0.2814)	0.7368 (0.1495)	0.7932 (0.1180)
cont	**0.0063** (0.0362)	**0.0069** (0.0116)	0.0073 (0.0172)	**0.8909** (0.2215)	0.6553 (0.3721)	0.7101 (0.2920)	0.7521 (0.1452)	**0.8303** (0.1195)
cont_{M}	0.0064 (0.0243)	0.0072 (0.0115)	**0.0070** (0.0145)	0.8878 (0.2294)	**0.6576** (0.3748)	**0.7123** (0.2923)	**0.7526** (0.1455)	0.8291 (0.1205)
2000								
true	0.0037 (0.0000)	0.0058 (0.0000)	0.0062 (0.0000)	0.9841 (0.0439)	0.8013 (0.2734)	0.8568 (0.2133)	0.8807 (0.1066)	0.9867 (0.0284)
HMM	0.2954 (0.3698)	0.2051 (0.3571)	0.2474 (0.3331)	0.8185 (0.2668)	0.3810 (0.4022)	0.5275 (0.4033)	0.5756 (0.1413)	0.7150 (0.2064)
discrete	0.0072 (0.0141)	0.0079 (0.0103)	0.0070 (0.0096)	**0.9487** (0.1520)	0.6156 (0.3863)	0.7107 (0.2520)	0.7584 (0.1549)	0.8173 (0.1177)
cont	**0.0042** (0.0101)	**0.0064** (0.0082)	0.0053 (0.0084)	0.9484 (0.1295)	**0.6676** (0.3558)	0.7270 (0.2534)	**0.7810** (0.1458)	**0.8614** (0.1131)
cont_{M}	0.0050 (0.0333)	0.0064 (0.0077)	**0.0060** (0.0181)	0.9480 (0.1376)	0.6638 (0.3608)	**0.7283** (0.2464)	0.7800 (0.1448)	0.8603 (0.1120)

Table 4: Mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 500, 1000, 2000$ from the daily three-state HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss (discrete, cont, and cont_{M}). The bold values represent the best parameter estimates and highest accuracy scores.

4.3 Misspecified Conditional Distributions

Rather than Markovian mixtures of Gaussian distributions, Bulla (2011) uses t-distributions for the state-conditions distributions in the HMMs. They demonstrate that the t-HMM is able to capture stylized facts of financial time series more effectively than the Gaussian HMM, particularly in terms of the slow decay of the autocorrelation function of absolute or squared returns. Moreover, the augmented mass around the mean and the fat tails of the t-distributions contribute to increased persistence in the estimated state sequence and enhance the model's robustness to outliers.

To examine the robustness of different models to misspecified conditional distributions, we generate observation sequences from t-HMMs, where each emission t-distribution has the same mean and covariance as specified in the daily parameter (30), with a degree of freedom of five, based on Bulla (2011)'s empirical estimates on daily return series of stock indices. The transition probability matrix of the state sequence remains unchanged.

Table 5 presents the mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 250, 500, 1000, 2000$ from the daily two-state t-HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss. In our experiments, both the true and estimated HMMs assume a Gaussian state-conditional distributions, allowing us to investigate the model's sensitivity to misspecification. Examining the parameter estimates, we observe a pronounced overestimation of transition probabilities in the HMMs compared to the well-specified scenario. The presence of fat tails in the t-distributions leads to an increased number of switches between the two states for the misspecified Gaussian HMM. In contrast, the jump models exhibit no difficulty in capturing the low values of the transition probabilities, thereby accurately capturing the persistence of the state sequence. In terms of BAC, we note a degradation of 5-8% for HMMs due to the misspecification of the conditional distributions, a 4% degradation for the discrete jump model, and only a 2% degradation for the continuous jump model. Consequently, the continuous jump model outperforms all other models in terms of both BAC and ROC-AUC for time series of at least 500 observations.

4.4 Misspecified Sojourn-Time Distributions

As another extension of HMMs, Bulla and Bulla (2006) propose hidden semi-Markov models (HSMM) to model financial return series. HSMM replaces the geometrically distributed sojourn time, implicitly assumed by the Markovian structure in HMMs, with a negative binomial distribution. By allowing for more flexible sojourn time distributions, they show that the model achieves improved performance in reproducing the long memory of return series.

To examine the models' performance under misspecified sojourn time distributions, we sample observation sequences from HSMMs. Following the approach in Nystrup et al. (2020a), we use shape parameters of $n_1 = 0.1$ for state 1 and $n_2 = 0.06$ for state 2 in the negative binomial distribution. These values are based on estimates from Bulla and Bulla (2006) on daily stock indices. We calibrate the other probability parameters in the negative binomial distribution to match the expected sojourn time of the HMM with the parameters in (30).

Table 6 presents the mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 250, 500, 1000, 2000$ from the daily two-state HSMM estimated with HMM, and the discrete and continuous jump models with and without mode loss. Due to the increased number of both very short and prolonged sojourns from the negative binomial distribution, the estimated HMM struggles to obtain reasonable estimates of the transition probabilities, and thus underperforms the jump models in terms of BAC and ROC-AUC. In contrast, the jump models exhibit better robustness to misspecified sojourn time distributions. The continuous jump model achieves a comparable BAC score to the discrete model, with a slight improvement in ROC-AUC.

5 An Application to the Nasdaq Index

In this application, we focus on the daily log-returns of the Nasdaq Composite index, a market-cap weighted index based on the approximately 3,000 companies listed on Nasdaq. We consider two distinct time periods: (i) 1996-2005 and (ii) 2013-2022, each spanning 10 years. By choosing these periods of

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC	ROC-AUC
250										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9839 (0.0576)	0.8073 (0.3069)	0.8956 (0.1511)	0.9766 (0.0805)
HMM	0.0005 (0.0085)	**-0.0005** (0.0181)	0.0070 (0.0041)	0.0142 (0.0065)	0.1942 (0.2333)	0.5392 (0.3739)	0.8597 (0.1889)	0.6921 (0.3165)	0.7759 (0.1710)	0.8426 (0.2059)
discrete	0.0005 (0.0026)	-0.0003 (0.0061)	**0.0082** (0.0041)	**0.0158** (0.0080)	0.0040 (0.0149)	0.0360 (0.0539)	**0.9272** (0.1472)	0.7039 (0.3413)	**0.8156** (0.1713)	0.8150 (0.1790)
cont	0.0006 (0.0012)	0.0001 (0.0026)	0.0083 (0.0031)	0.0124 (0.0050)	0.0031 (0.0058)	**0.0157** (0.0261)	0.8799 (0.1649)	**0.7511** (0.3391)	0.8155 (0.1823)	**0.9036** (0.1833)
cont$_\mathrm{M}$	**0.0006** (0.0010)	-0.0000 (0.0030)	0.0082 (0.0022)	0.0133 (0.0051)	**0.0026** (0.0051)	0.0158 (0.0295)	0.9001 (0.1630)	0.7211 (0.3609)	0.8106 (0.1906)	0.8940 (0.1852)
500										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9877 (0.0228)	0.8330 (0.2559)	0.9103 (0.1269)	0.9848 (0.0448)
HMM	0.0006 (0.0007)	-0.0000 (0.0141)	0.0067 (0.0027)	0.0164 (0.0054)	0.1194 (0.1434)	0.4191 (0.3545)	0.9191 (0.1279)	0.7086 (0.2838)	0.8139 (0.1549)	0.8955 (0.1590)
discrete	0.0006 (0.0009)	-0.0002 (0.0060)	0.0080 (0.0022)	**0.0169** (0.0076)	0.0025 (0.0057)	0.0313 (0.0453)	**0.9507** (0.0983)	0.7390 (0.3112)	0.8448 (0.1634)	0.8550 (0.1604)
cont	0.0006 (0.0005)	-0.0002 (0.0024)	0.0080 (0.0008)	0.0141 (0.0049)	**0.0023** (0.0019)	**0.0162** (0.0307)	0.9157 (0.1254)	**0.8023** (0.2784)	0.8590 (0.1600)	**0.9285** (0.1565)
cont$_\mathrm{M}$	**0.0006** (0.0005)	**-0.0003** (0.0029)	**0.0079** (0.0008)	0.0146 (0.0052)	0.0027 (0.0019)	0.0201 (0.0265)	0.9262 (0.1091)	0.7963 (0.2708)	**0.8613** (0.1536)	0.9219 (0.1496)
1000										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9882 (0.0142)	0.8669 (0.1956)	0.9276 (0.0970)	0.9875 (0.0478)
HMM	**0.0006** (0.0003)	-0.0004 (0.0171)	0.0067 (0.0006)	**0.0174** (0.0037)	0.0622 (0.0830)	0.2857 (0.3090)	0.9587 (0.0471)	0.7425 (0.2672)	0.8506 (0.1380)	0.9303 (0.0994)
discrete	0.0006 (0.0003)	**-0.0006** (0.0062)	**0.0079** (0.0006)	0.0189 (0.0093)	0.0018 (0.0011)	0.0278 (0.0404)	**0.9827** (0.0354)	0.7635 (0.2735)	0.8731 (0.1370)	0.8748 (0.1362)
cont	0.0006 (0.0003)	-0.0006 (0.0028)	0.0079 (0.0005)	0.0168 (0.0061)	**0.0022** (0.0014)	**0.0169** (0.0175)	0.9615 (0.0811)	**0.8201** (0.2282)	0.8908 (0.1254)	**0.9513** (0.1123)
cont$_\mathrm{M}$	0.0006 (0.0003)	-0.0006 (0.0030)	0.0079 (0.0005)	0.0170 (0.0062)	0.0020 (0.0012)	0.0179 (0.0195)	0.9699 (0.0631)	0.8127 (0.2349)	**0.8913** (0.1238)	0.9398 (0.1178)
2000										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9885 (0.0098)	0.8902 (0.1230)	0.9394 (0.0608)	0.9911 (0.0297)
HMM	**0.0006** (0.0002)	-0.0003 (0.0153)	0.0068 (0.0004)	0.0178 (0.0025)	0.0418 (0.0449)	0.2227 (0.2352)	0.9697 (0.0199)	0.7491 (0.2334)	0.8594 (0.1195)	0.9421 (0.0734)
discrete	0.0006 (0.0002)	-0.0009 (0.0047)	**0.0079** (0.0004)	0.0189 (0.0082)	0.0016 (0.0008)	0.0185 (0.0241)	**0.9894** (0.0124)	0.7955 (0.2112)	0.8925 (0.1045)	0.8926 (0.1044)
cont	0.0006 (0.0002)	**-0.0008** (0.0026)	0.0079 (0.0003)	**0.0176** (0.0055)	**0.0018** (0.0009)	**0.0144** (0.0126)	0.9842 (0.0320)	**0.8434** (0.1639)	**0.9138** (0.0837)	**0.9655** (0.0731)
cont$_\mathrm{M}$	0.0006 (0.0002)	-0.0009 (0.0026)	0.0079 (0.0003)	0.0177 (0.0053)	0.0017 (0.0009)	0.0145 (0.0126)	0.9858 (0.0287)	0.8386 (0.1664)	0.9122 (0.0851)	0.9550 (0.0791)

Table 5: Mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 250, 500, 1000, 2000$ from the daily two-state t-HMM estimated with HMM, and the discrete and continuous jump models with and without mode loss (discrete, cont, and cont$_\mathrm{M}$). The bold values represent the best parameter estimates and highest accuracy scores.

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC	ROC-AUC
250										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9451 (0.2083)	0.7223 (0.4123)	0.8337 (0.1932)	0.9075 (0.1849)
HMM	0.0002 (0.0076)	-0.0014 (0.0083)	**0.0084** (0.0032)	0.0102 (0.0059)	0.2760 (0.3198)	0.4715 (0.3771)	0.7870 (0.2727)	0.6767 (0.3512)	0.7318 (0.1967)	0.7303 (0.2834)
discrete	**0.0005** (0.0016)	**-0.0002** (0.0026)	0.0086 (0.0028)	**0.0138** (0.0045)	0.0036 (0.0073)	0.0165 (0.0275)	**0.8843** (0.2252)	**0.7281** (0.3634)	**0.8062** (0.1981)	0.7811 (0.2213)
cont	0.0004 (0.0017)	0.0001 (0.0020)	0.0094 (0.0036)	0.0118 (0.0045)	0.0037 (0.0063)	**0.0113** (0.0186)	0.8389 (0.2254)	0.7191 (0.3682)	0.7790 (0.2126)	**0.8120** (0.2542)
cont_{M}	0.0005 (0.0013)	-0.0001 (0.0020)	0.0089 (0.0031)	0.0129 (0.0045)	**0.0034** (0.0067)	0.0109 (0.0200)	0.8600 (0.2280)	0.7212 (0.3743)	0.7906 (0.2068)	0.8083 (0.2447)
500										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9565 (0.1785)	0.7270 (0.4029)	0.8418 (0.1993)	0.9316 (0.1566)
HMM	0.0000 (0.0059)	**-0.0006** (0.0068)	**0.0084** (0.0027)	0.0119 (0.0057)	0.2256 (0.2887)	0.3892 (0.3775)	0.8295 (0.2536)	0.7012 (0.3521)	0.7654 (0.2212)	0.7727 (0.3019)
discrete	**0.0005** (0.0015)	-0.0003 (0.0024)	0.0086 (0.0027)	**0.0142** (0.0044)	0.0033 (0.0083)	0.0163 (0.0263)	**0.9046** (0.1952)	0.7554 (0.3482)	**0.8300** (0.1917)	0.8280 (0.2004)
cont	0.0004 (0.0013)	-0.0000 (0.0018)	0.0090 (0.0031)	0.0128 (0.0045)	**0.0031** (0.0060)	**0.0120** (0.0240)	0.8680 (0.1988)	0.7593 (0.3470)	0.8137 (0.2051)	**0.8600** (0.2238)
cont_{M}	0.0005 (0.0015)	-0.0001 (0.0020)	0.0089 (0.0031)	0.0129 (0.0046)	0.0047 (0.0123)	0.0146 (0.0216)	0.8642 (0.2009)	**0.7771** (0.3298)	0.8207 (0.2021)	0.8559 (0.2216)
1000										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9674 (0.1578)	0.7251 (0.3988)	0.8463 (0.2009)	0.9454 (0.1384)
HMM	**0.0006** (0.0051)	0.0001 (0.0060)	0.0083 (0.0024)	0.0126 (0.0055)	0.1949 (0.2892)	0.3448 (0.3766)	0.8482 (0.2642)	0.7361 (0.3499)	0.7921 (0.2199)	0.8112 (0.2782)
discrete	0.0005 (0.0011)	**-0.0004** (0.0022)	**0.0082** (0.0022)	**0.0143** (0.0043)	0.0019 (0.0048)	0.0157 (0.0262)	**0.9230** (0.1761)	0.7653 (0.3454)	**0.8442** (0.2011)	0.8465 (0.2046)
cont	0.0005 (0.0009)	-0.0003 (0.0017)	0.0084 (0.0022)	0.0133 (0.0044)	**0.0020** (0.0042)	0.0106 (0.0161)	0.8939 (0.1772)	0.7704 (0.3471)	0.8322 (0.2104)	**0.8713** (0.2201)
cont_{M}	0.0005 (0.0009)	-0.0003 (0.0018)	0.0084 (0.0022)	0.0134 (0.0045)	0.0023 (0.0038)	**0.0124** (0.0179)	0.8898 (0.1869)	**0.7803** (0.3329)	0.8350 (0.2119)	0.8657 (0.2229)
2000										
true	0.0006 (0.0000)	-0.0008 (0.0000)	0.0078 (0.0000)	0.0174 (0.0000)	0.0021 (0.0000)	0.0120 (0.0000)	0.9894 (0.0717)	0.7689 (0.3660)	0.8792 (0.1831)	0.9588 (0.1250)
HMM	0.0002 (0.0040)	**-0.0005** (0.0044)	0.0081 (0.0018)	0.0136 (0.0051)	0.1429 (0.2447)	0.2602 (0.3489)	0.8950 (0.2141)	0.7505 (0.3509)	0.8227 (0.2212)	0.8319 (0.2754)
discrete	**0.0006** (0.0007)	-0.0004 (0.0020)	**0.0080** (0.0013)	**0.0149** (0.0040)	0.0012 (0.0028)	0.0126 (0.0185)	**0.9617** (0.1017)	0.7751 (0.3412)	**0.8684** (0.1870)	0.8714 (0.1868)
cont	0.0006 (0.0005)	-0.0004 (0.0015)	0.0080 (0.0013)	0.0143 (0.0042)	0.0019 (0.0022)	0.0096 (0.0116)	0.9082 (0.1538)	0.8128 (0.3048)	0.8605 (0.2017)	0.8958 (0.2092)
cont_{M}	0.0006 (0.0006)	-0.0004 (0.0016)	0.0080 (0.0014)	0.0144 (0.0042)	**0.0023** (0.0030)	**0.0125** (0.0144)	0.9176 (0.1496)	**0.8178** (0.2939)	0.8677 (0.1885)	**0.8964** (0.1922)

Table 6: Mean and standard deviation (in parenthesis) of parameter estimates and accuracy scores of 1024 simulations of length $T = 250, 500, 1000, 2000$ from the daily two-state HSMM estimated with HMM, and the discrete and continuous jump models with and without mode loss (discrete, cont, and cont_{M}). The bold values represent the best parameter estimates and highest accuracy scores.

contrasting market conditions, with the former encompassing the rise and burst of the dot-com bubble and the latter representing a relatively calm period, we aim to gain insights into the performance of the models under different market conditions. Considering the lack of persistence of HMMs in predicting regimes in the equity market shown in previous studies (Nystrup et al., 2020a,b), we opt to explore the three jump models in this application. To select the jump penalty parameter λ, we leverage the optimal values derived from the simulation study in Section 4.1 on daily two-state HMMs. Specifically, for the discrete jump model, we choose $\lambda = 10^2$, while for the continuous jump models, we select $\lambda = 10^3$.

5.1 The 1996-2005 Time Period

Table 7 presents the estimated parameters for the three jump models with two states applied to the Nasdaq Composite index from 1996 through 2005. The parameters include the mean return and volatility under each regime, and the resulting empirical transition probabilities. Consistent with previous literature, we observe that the first and second states are characterized by positive mean return and low volatility, and negative mean return high volatility, respectively. Thus, we interpret the first state as the bull regime and the second state as the bear regime.

We observe that the estimated mean returns and volatilities are similar for the three models. However, the continuous jump model yields higher estimates of the transition probabilities compared to the discrete model. This stems from the continuous model's ability to accurately capture smooth transitions in the estimated probabilities. We emphasize that the estimated transition probabilities from all three models are significantly lower than those reported in previous articles that consider broader equity indexes (Hardy, 2001; Nystrup et al., 2020b). This is attributed to the composition of the Nasdaq index, which is predominantly comprised of technology companies.

	μ_0	μ_1	σ_0	σ_1	γ_{01}	γ_{10}
discrete	0.1148%	-0.1528%	1.2160%	2.6879%	0.0012	0.0025
cont	0.1029%	-0.1879%	1.2891%	2.8616%	0.0021	0.0063
cont_M	0.0995%	-0.1750%	1.2862%	2.8549%	0.0021	0.0062

Table 7: Estimated mean return and volatility under each of the two regimes and the transition probabilities on the daily log-returns of the Nasdaq Index from 1996 to 2005.

Figure 2 displays the regimes estimated by the three jump models. Notably, the discrete jump model identifies the dot-com crash as a single bear period, while the continuous jump models are able to detect two rebound periods. The ability to capture rebounds is a result of the smooth transition within the probability simplex in the continuous models. We underscore that the mode loss penalty, when included, effectively smooths the probability curve, thereby eliminating minor fluctuations with amplitudes lower than approximately 30%. This smoothing results in more persistent state probabilities. In certain applications, maintaining unchanged probability estimation can eliminate the necessity for portfolio rebalancing, ultimately leading to decreased turnover and reduced transaction costs.

5.2 The 2013-2022 Time Period

Table 8 presents the estimated parameters for the three jump model from 2013 to 2022. The estimated values of the transition probabilities provide support for that the continuous jump models yield slightly higher estimates due to their smoother state transitions. Moreover, the inclusion of the mode loss penalty appears to further enhance the persistence of the estimated probabilities. Similarly to the 1996-2005, we interpret the first and second states as bull and bear regimes, respectively.

Figure 3 depicts the behavior of the three jump models during the relatively calm market period from 2013 to 2022. The discrete jump model successfully identifies the three major bear market periods: in 2019, the 2020 COVID-19 pandemic, and the Fed's interest rate hikes in 2022. However, the continuous jump models also capture other market downturns during this period, e.g. the drawdown around 2016. This ability to detect and respond to smaller market downturns can aid in active risk management even in periods

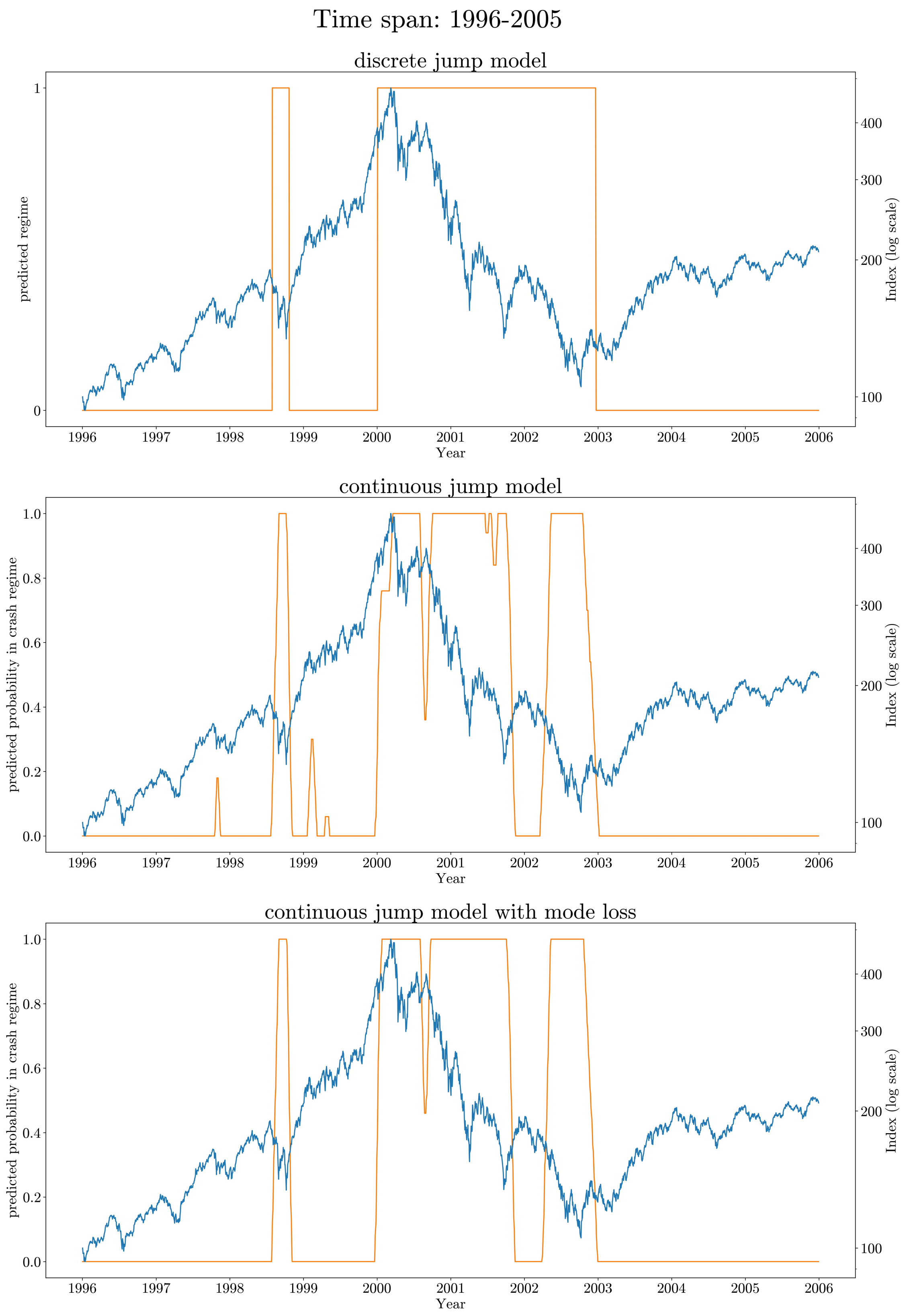

Figure 2: Regimes (in yellow) of the Nasdaq Index (in blue) from 1996 to 2005 as estimated by the three jump models.

	μ_0	μ_1	σ_0	σ_1	γ_{01}	γ_{10}
discrete	0.0829%	-0.1336%	0.9828%	2.3885%	0.0014	0.0051
cont	0.0903%	-0.1383%	0.9531%	2.3065%	0.0029	0.0117
cont_M	0.0868%	-0.1386%	0.9686%	2.3490%	0.0019	0.0077

Table 8: Estimated mean return and volatility under each of the two regimes and the transition probabilities on the daily log-returns of the Nasdaq Index from 2013 to 2022.

of relative market stability and highlights the advantage of the continuous jump model in identifying varying market conditions.

6 Conclusion

In this article, we extend the statistical jump model Nystrup et al. (2020b,a) by generalizing the discrete hidden state variable into a probability vector over all regimes. The new model, referred to as a continuous jump model, not only enables the estimation of regime probabilities but also enhances robustness over the original discrete model by smoothly transitioning from one regime to another. We provided a probabilistic interpretation of the new model and demonstrated its advantages through simulations and real-world data experiments. Specifically, our simulation studies and an application to the Nasdaq Index demonstrate that the continuous jump model offers advantages over competing methods such as hidden Markov models and discrete jump models. The advantages of the new approach become particularly pronounced when dealing with relatively short, imbalanced, and highly persistent time series, making it well-suited for a broad range of applications in finance such as regime-aware portfolio models and risk management.

While not used in this article, it is worth noting that the continuous jump model can be readily augmented with feature selection, hyperparameter tuning, and model selection techniques, similar to those discussed in Nystrup et al. (2021); Cortese et al. (2023b).

References

Aminikhanghahi, S. and Cook, D. J. (2017). A survey of methods for time series change point detection. *Knowledge and information systems*, 51(2):339–367.

Ang, A. and Timmermann, A. (2012). Regime changes and financial markets. *Annual Review of Financial Economics*, 4(1):313–337.

Arthur, D. and Vassilvitskii, S. (2007). K-means++: The advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, USA. Society for Industrial and Applied Mathematics.

Attouch, H., Bolte, J., Redont, P., and Soubeyran, A. (2010). Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Lojasiewicz inequality. *Mathematics of Operations Research*, 35(2):438–457.

Bae, G. I., Kim, W. C., and Mulvey, J. M. (2014). Dynamic asset allocation for varied financial markets under regime switching framework. *European Journal of Operational Research*, 234(2):450–458. 60 years following Harry Markowitz's contribution to portfolio theory and operations research.

Balakrishnan, S., Wainwright, M. J., and Yu, B. (2017). Statistical guarantees for the EM algorithm: From population to sample-based analysis. *The Annals of Statistics*, 45(1):77 – 120.

Baum, L., Petrie, T., Soules, G., and Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, pages 164–171.

Bemporad, A., Breschi, V., Piga, D., and Boyd, S. P. (2018). Fitting jump models. *Automatica*, 96:11–21.

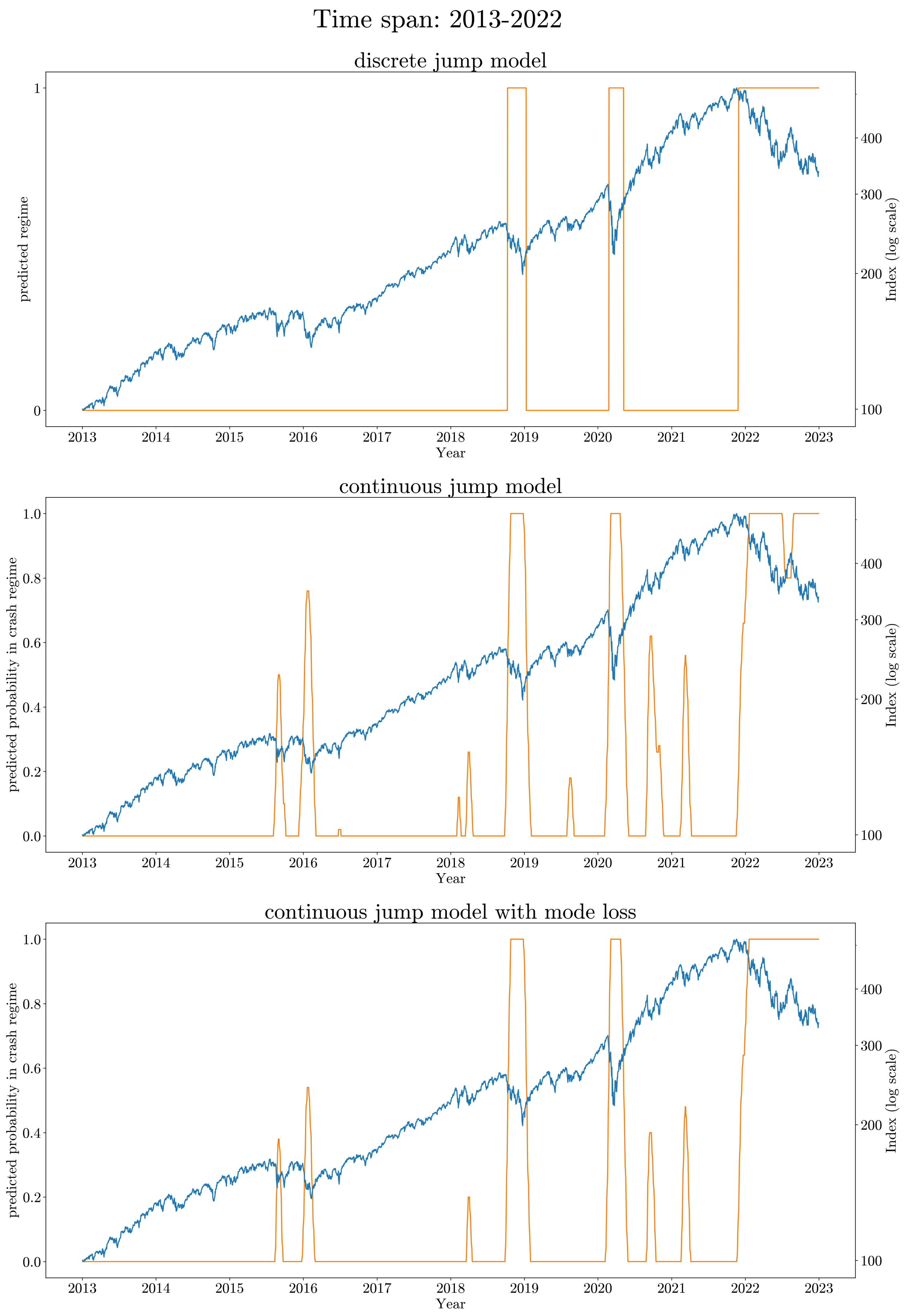

Figure 3: Regimes (in yellow) of the Nasdaq Index (in blue) from 2013 to 2022 as estimated by the three jump models.

Bertsekas, D. P. (1999). *Nonlinear Programming.* Athena Scientific, Belmont, 2nd edition.

Bertsimas, D. and Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization.* Athena Scientific.

Bickel, P. J., Ritov, Y., and Rydén, T. (1998). Asymptotic normality of the maximum-likelihood estimator for general hidden Markov models. *The Annals of Statistics*, 26(4):1614 – 1635.

Bolte, J., Sabach, S., and Teboulle, M. (2014). Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494.

Bradley, A. P. (1997). The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159.

Brodersen, K. H., Ong, C. S., Stephan, K. E., and Buhmann, J. M. (2010). The balanced accuracy and its posterior distribution. In *2010 20th International Conference on Pattern Recognition*, pages 3121–3124.

Bry, G. and Boschan, C. (1971). *Cyclical Analysis of Time Series: Selected Procedures and Computer Programs.* NBER.

Bulla, J. (2011). Hidden Markov models with t components: Increased persistence and other aspects. *Quantitative Finance*, 11(3):459–475.

Bulla, J. and Berzel, A. (2008). Computational issues in parameter estimation for stationary hidden Markov models. *Computational Statistics*, 23(1):1–18.

Bulla, J. and Bulla, I. (2006). Stylized facts of financial time series and hidden semi-Markov models. *Computational Statistics & Data Analysis*, 51(4):2192–2209. Nonlinear Modelling and Financial Econometrics.

Bulla, J., Mergner, S., Bulla, I., Sesboüé, A., and Chesneau, C. (2011). Markov-switching asset allocation: Do profitable strategies exist? *Journal of Asset Management*, 12(4):310–321.

Cartea, A. and Jaimungal, S. (2013). Modelling asset prices for algorithmic and high-frequency trading. *Applied Mathematical Finance*, 20(6):512–547.

Cortese, F., Kolm, P., and Lindström, E. (2023a). What drives cryptocurrency returns? a sparse statistical jump model approach. *Digit Finance.*

Cortese, F. P., Kolm, P. N., and Lindström, E. (2023b). Generalized information criteria for sparse statistical jump models. In Linde, P., editor, *Symposium i anvendt statistik, Vol 44*, Copenhagen. Copenhagen Business School.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *J. R. Stat. Soc. B*, pages 1–38.

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

Durbin, R., Eddy, S. R., Krogh, A., and Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids.* Cambridge University Press.

Ebbers, J., Heymann, J., Drude, L., Glarner, T., Haeb-Umbach, R., and Raj, B. (2017). Hidden Markov model variational autoencoder for acoustic unit discovery. In *InterSpeech*, pages 488–492.

Elliott, R. J., Siu, T. K., and Badescu, A. (2010). On mean-variance portfolio selection under a hidden Markovian regime-switching model. *Economic Modelling*, 27(3):678–686.

Fine, S., Singer, Y., and Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32(1):41–62.

Gales, M. and Young, S. (2007). The application of hidden Markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304.

Ghahramani, Z. and Jordan, M. (1995). Factorial hidden Markov models. In Touretzky, D., Mozer, M., and Hasselmo, M., editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press.

Goh, C., Dauwels, J., Mitrovic, N., Asif, M. T., Oran, A., and Jaillet, P. (2012). Online map-matching based on hidden Markov model for real-time traffic sensing applications. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 776–781.

Goutte, S., Ismail, A., and Pham, H. (2017). Regime-switching stochastic volatility model: estimation and calibration to VIX options. *Applied Mathematical Finance*, 24(1):38–75.

Gray, S. F. (1996). Modeling the conditional distribution of interest rates as a regime-switching process. *Journal of Financial Economics*, 42(1):27–62.

Gu, J. and Mulvey, J. M. (2021). Factor momentum and regime-switching overlay strategy. *The Journal of Financial Data Science*, 3(4):101–129.

Hallac, D., Vare, S., Boyd, S., and Leskovec, J. (2017). Toeplitz inverse covariance-based clustering of multivariate time series data. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, pages 215–223, New York, NY, USA. Association for Computing Machinery.

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57(2):357–384.

Hamilton, J. D. (2010). Regime switching models. In Durlauf, S. N. and Blume, L. E., editors, *Macroeconometrics and Time Series Analysis*, The New Palgrave Economics Collection. Palgrave Macmillan, London.

Hamilton, J. D. and Susmel, R. (1994). Autoregressive conditional heteroskedasticity and changes in regime. *Journal of Econometrics*, 64(1):307–333.

Hand, D. and Till, R. (2001). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45(2):171–186.

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

Himberg, J., Korpiaho, K., Mannila, H., Tikanmaki, J., and Toivonen, H. T. (2001). Time series segmentation for context recognition in mobile devices. In *Proceedings 2001 IEEE international conference on data mining*, pages 203–210. IEEE.

Hsu, D., Kakade, S. M., and Zhang, T. (2012). A spectral algorithm for learning hidden Markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480. JCSS Special Issue: Cloud Computing 2011.

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

Kowalski, M. (2009). Sparse regression using mixed norms. *Applied and Computational Harmonic Analysis*, 27(3):303–324.

Levin, D. A., Peres, Y., and Wilmer, E. L. (2017). *Markov Chains and Mixing Times.* American Mathematical Society, 2nd edition.

Li, X. and Mulvey, J. M. (2021). Portfolio optimization under regime switching and transaction costs: Combining neural networks and dynamic programs. *INFORMS Journal on Optimization*, 3(4):398–417.

Li, X. and Mulvey, J. M. (2023). Optimal portfolio execution in a regime-switching market with non-linear impact costs: Combining dynamic program and neural network. *pre-print.*

Lin, M. (2023). *Essays on Applications of Networks and Discrete Optimization.* Ph.d. dissertation, Princeton University.

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

Mamon, R. S. and Elliott, R. J., editors (2007). *Hidden Markov Models in Finance.* Springer, New York, NY.

Mulvey, J. M. and Liu, H. (2016). Identifying economic regimes: Reducing downside risks for university endowments and foundations. *The Journal of Portfolio Management*, 43(1):100–108.

Munkres, J. (2000). *Topology.* Pearson, 2nd edition.

Ng, A., Jordan, M., and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14.

Nystrup, P., Kolm, P. N., and Lindström, E. (2020a). Greedy online classification of persistent market states using realized intraday volatility features. *The Journal of Financial Data Science*, 2(3):25–39.

Nystrup, P., Kolm, P. N., and Lindström, E. (2021). Feature selection in jump models. *Expert Systems with Applications*, 184:115558.

Nystrup, P., Lindström, E., and Madsen, H. (2020b). Learning hidden Markov models with persistent states by penalizing jumps. *Expert Systems with Applications*, 150:113307.

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

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

Pagan, A. R. and Sossounov, K. A. (2003). A simple framework for analysing bull and bear markets. *Journal of Applied Econometrics*, 18(1):23–46.

Peyré, G. and Cuturi, M. (2019). Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5-6):355–607.

Picard, F., Lebarbier, E., Budinskà, E., and Robin, S. (2011). Joint segmentation of multivariate Gaussian processes using mixed linear models. *Computational Statistics & Data Analysis*, 55(2):1160–1170.

Qi, Y. and Ishak, S. (2014). A hidden Markov model for short term prediction of traffic conditions on freeways. *Transportation Research Part C: Emerging Technologies*, 43:95–111. Special Issue on Short-term Traffic Flow Forecasting.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Reus, L. and Mulvey, J. M. (2016). Dynamic allocations for currency futures under switching regimes signals. *European Journal of Operational Research*, 253(1):85–93.

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

Sawhney, A. (2020). Regime identification, curse of dimensionality and deep generative models. Technical report, Quantitative Brokers.

Schwert, G. W. (1989). Why does stock market volatility change over time? *The Journal of Finance*, 44(5):1115–1153.

Stock, J. H. and Watson, M. W. (1996). Evidence on structural instability in macroeconomic time series relations. *Journal of Business & Economic Statistics*, 14(1):11–30.

Uysal, A. S. and Mulvey, J. M. (2021). A machine learning approach in regime-switching risk parity portfolios. *The Journal of Financial Data Science*, 3(2):87–108.

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

Witten, D. M. and Tibshirani, R. (2010). A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726. PMID: 20811510.

Wright, S. J. (2015). Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34.

Wu, C. F. J. (1983). On the convergence properties of the EM algorithm. *The Annals of Statistics*, 11(1):95 – 103.

Yang, F., Balakrishnan, S., and Wainwright, M. J. (2017). Statistical and computational guarantees for the Baum-Welch algorithm. *The Journal of Machine Learning Research*, 18(1):4528–4580.

Zhang, M., Jiang, X., Fang, Z., Zeng, Y., and Xu, K. (2019). High-order hidden Markov model for trend prediction in financial time series. *Physica A: Statistical Mechanics and its Applications*, 517:1–12.

Zheng, K., Li, Y., and Xu, W. (2021). Regime switching model estimation: Spectral clustering hidden Markov model. *Annals of Operations Research*, 303:297–319.

A Appendix: Proofs

Proof of Proposition 1. From the probabilistic assumptions, the joint likelihood is

$$p(\boldsymbol{Y}, \boldsymbol{S}|\Theta) = p(\boldsymbol{Y}|\boldsymbol{S}, \Theta)p(\boldsymbol{S}) \tag{33}$$

$$= \prod_{t=0}^{T-1} p(\boldsymbol{y}_t|\Theta, \boldsymbol{s}_t) \times \pi(\boldsymbol{s}_0) \times \prod_{t=1}^{T-1} K\left(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t\right) . \tag{34}$$

Thus

$$\log p(\boldsymbol{Y}, \boldsymbol{S}|\Theta) = \sum_{t=0}^{T-1} \log p(\boldsymbol{y}_t|\Theta, \boldsymbol{s}_t) + \log \pi(\boldsymbol{s}_0) + \sum_{t=1}^{T-1} \log K\left(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t\right) . \tag{35}$$

For the first term, by Jensen's inequality we have

$$\log p(\boldsymbol{y}_t|\Theta, \boldsymbol{s}_t) = \log \left(\sum_{k=0}^{K-1} s_{tk} p(\boldsymbol{y}_t|\boldsymbol{\theta}_k) \right) \tag{36}$$

$$\geq \sum_{k=0}^{K-1} s_{tk} \log p(\boldsymbol{y}_t|\boldsymbol{\theta}_k) . \tag{37}$$

Inserting (37) into (35), we obtain

$$\log p(\boldsymbol{Y}, \boldsymbol{S}|\Theta) \geq \sum_{t,k} s_{tk} \log p(\boldsymbol{y}_t|\boldsymbol{\theta}_k) + \log \pi(\boldsymbol{s}_0) + \sum_{t=1}^{T-1} \log K\left(\boldsymbol{s}_{t-1}, \boldsymbol{s}_t\right) . \tag{38}$$

From equation (17), it follows that the above right hand side is precisely the negation of the objective function J. Therefore minimizing J is equivalent to maximizing a lower bound of the joint log-likelihood function. □

Proof of Proposition 2. From the previous proof, we know that

$$\log p(\boldsymbol{Y}, \boldsymbol{S}|\Theta) = \sum_{t=0}^{T-1} \log p(\boldsymbol{y}_t|\Theta, \boldsymbol{s}_t) + \log p(\boldsymbol{S}) \tag{39}$$

$$\geq \sum_{t,k} s_{tk} \log p(\boldsymbol{y}_t|\boldsymbol{\theta}_k) + \log p(\boldsymbol{S}) \tag{40}$$

Inserting (18) into (40), we obtain

$$\sum_{t,k} s_{t,k} \left(-l(\boldsymbol{y}_t, \boldsymbol{\theta}_k) - \log \nu\right) - \mathcal{L}(\boldsymbol{S}) - \log \eta = - \left(\sum_{t,k} s_{t,k} l(\boldsymbol{y}_t, \boldsymbol{\theta}_k) + \mathcal{L}(\boldsymbol{S}) \right) + \text{const} \tag{41}$$

$$= - J + \text{const} . \tag{42}$$

Thus, minimizing J is equivalent to maximizing a lower bound of the log-likelihood $p(\boldsymbol{Y}, \boldsymbol{S}|\Theta)$. □