Greedy Online Classification of Persistent Market States Using Realized Intraday Volatility Features

Nystrup, Peter; Kolm, Petter N.; Lindström, Erik

Published in:
Journal of Financial Data Science

Link to article, DOI:
10.3905/jfds.2020.2.3.025

Publication date:
2020

Document Version
Peer reviewed version

GREEDY ONLINE CLASSIFICATION OF PERSISTENT MARKET STATES USING REALIZED INTRADAY VOLATILITY FEATURES

Peter Nystrup
Centre for Mathematical Sciences
Lund University
Lund, Sweden
Department of Applied Mathematics and Computer Science
Technical University of Denmark
Kgs. Lyngby, Denmark.
peter.nystrup@matstat.lu.se

Petter N. Kolm
Courant Institute of Mathematical Sciences
New York University
New York, USA
petter.kolm@nyu.edu

Erik Lindström
Centre for Mathematical Sciences
Lund University
Lund, Sweden
erik.lindstrom@matstat.lu.se

May 6, 2020

ABSTRACT

In many financial applications it is important to classify time series data without any latency while maintaining persistence in the identified states. We propose a greedy online classifier that contemporaneously determines which hidden state a new observation belongs to without the need to parse historical observations and without compromising persistence. Our classifier is based on the idea of clustering temporal features while explicitly penalizing jumps between states by a fixed-cost regularization term that can be calibrated to achieve a desired level of persistence. Through a series of return simulations, we show that in most settings our new classifier remarkably obtains a higher accuracy than the correctly specified maximum likelihood estimator. We illustrate that the new classifier is more robust to misspecification and yields state sequences that are significantly more persistent both in and out of sample. We demonstrate how classification accuracy can be further improved by including features that are based on intraday data. Finally, we apply the new classifier to estimate persistent states of the S&P 500 index.

Introduction

The hidden Markov model (HMM) is popular in many areas with applications ranging from facial recognition (Nefian and Hayes, 1998) and natural language processing (Gales and Young, 2008, Kang et al., 2018) to wind- and solar-power forecasting (Pinson et al., 2008, Bhardwaj et al., 2013), fraud detection (Robinson and Aria, 2018), and risk and return modeling in finance and energy markets (Dias and Ramos, 2014, Petropoulos et al., 2016, Nystrup et al., 2018).

In many financial applications it is highly desirable that the state sequence has a certain level of persistence. For example, if the state sequence is used as the basis of an asset allocation or trading strategy, then lower persistence leads to higher turnover and transaction costs and, consequently, poorer returns (Nystrup et al., 2015a). If the model is misspecified (e.g., because the Markov assumption or the conditional distributions are incorrect) or not properly estimated (e.g., due to limited data being available, high dimensionality, nonstationarity, or failure to converge to the global optimum in the estimation process), typically it leads to unstable and impersistent estimates of the underlying state sequence and the introduction of unnecessary states (Bulla, 2011, Fox et al., 2011, Fiecas et al., 2017). In

particular, when the state sequence contains far too many jumps the model reduces to a mixture model with no temporal information, which is of limited use in practice.

In this paper we propose a greedy online classifier that contemporaneously determines which hidden state a new observation belongs to without the need to parse historical observations and without compromising persistence. Online classification is critical for applications such as asset allocation, patient monitoring, intrusion detection, and credit card fraud monitoring where delay following the onset of a suspicious activity might be detrimental (Nystrup et al., 2017a, Garg and Pichkhadze, 2019). When classifying observations online without looking at later observations it is considerably harder to obtain persistent state estimates because of the inability to apply smoothing.

Our greedy classifier is based on the idea of clustering temporal features while explicitly penalizing jumps between states by a fixed-cost regularization term that can be calibrated to achieve a desired level of persistence. The starting point is a persistent jump model estimated using the framework proposed by Bemporad et al. (2018). Our jump model estimator builds on the work of Zheng et al. (2019) and Nystrup et al. (2020b), incorporating a new feature set for learning HMMs that is entirely backward looking and therefore can be applied in a truly online fashion.

Through a series of simulations, we show that in most settings our new jump estimator remarkably obtains a higher accuracy than the correctly specified maximum likelihood estimator, when using the same initial parameters for the estimate. At the same time the inferred state sequences are significantly more persistent both in and out of sample. Moreover, we demonstrate how the greedy classifier's accuracy can be further improved by including features that are based on intraday data. Finally, we compare two versions of our jump estimator to an HMM on the S&P500 index, where the differences in persistence are striking. While the HMM detects state changes equally quickly as the jump model, it comes at the cost of many spurious and short-lived changes. Consequently, if these estimated states were used as part of a regime-based asset allocation strategy, the lower persistence would lead to higher turnover and transaction costs.

Related work

Time series segmentation. Given a sequence of time series data, it is often desirable to partition it into states or segments, where each segment can be explained by an as simple model as possible. Variants of the problem have been studied in several contexts, including change-point detection (Nystrup et al., 2016), segmentation (Hallac et al., 2019), trend filtering (Kim et al., 2009), and mixture models (Verbeek et al., 2003, Samé et al., 2011). The task of segmenting a time series is different from clustering in general, because the temporal ordering of the data must be taken into account.

Model extensions. The classical, two-state Gaussian HMM has been extended in numerous ways to overcome its shortcomings in specific applications. Through conditional distributions with heavier tails (Bulla, 2011) or time-varying parameters (Nystrup et al., 2017b) it is possible to increase persistence and improve the model's ability to reproduce long memory. Other studies have relaxed the Markov assumption on sojourn-time distributions (Bulla and Bulla, 2006, Langrock and Zucchini, 2011), considered higher-order Markov chains (Shamshad et al., 2005), and increased the size of the hidden state space (Ghahramani and Jordan, 1997, Nystrup et al., 2015b). In Bayesian settings the level of persistence can be adjusted through a hyperparameter in the hierarchical Dirichlet prior distribution imposed on the transition probability matrix (Beal et al., 2002, Fox et al., 2011).

Estimation. Standard approaches to estimating HMMs include direct numerical maximization of the likelihood function, the Expectation–Maximization (EM) algorithm (Baum et al., 1970, Dempster et al., 1977), and Bayesian estimation using Markov chain Monte Carlo methods (Rydén, 2008). The maximum likelihood estimator has favorable properties as the number of observations tends to infinity. In particular, it is consistent and asymptotically efficient. However, computationally it is not as tractable (Andersson et al., 2003).

Spectral methods. Rousseeuw et al. (2015) used the spectral clustering algorithm of Ng et al. (2002) to estimate the parameters of a discrete-output HMM. They were able to recover the hidden partition without prior knowledge of the specific model parameters. Subsequently, Zheng et al. (2019) used spectral clustering to learn a continuous-output HMM based on simple features constructed from the observed time series. Their results showed that spectral clustering can be more robust and more accurate than maximum likelihood estimation when the persistence in the state process is low. Nevertheless, both methods have difficulties with the highly persistent processes that are characteristic for many financial time series (Rydén et al., 1998, Cont, 2001, Nystrup et al., 2017b).

Jump penalization. Nystrup et al. (2020b) proposed a framework for fitting persistent jump models by combining the explicit penalization approach of Bemporad et al. (2018) with the temporal features for learning HMMs proposed

by Zheng et al. (2019). The jump penalty regularizes the estimation problem and provides control over the transition rate. Unfortunately, the forward-looking nature of some of the features in this approach make them less useful in practical applications where the future is unknown and observations are processed in an online fashion as they arrive.

Online state classification. In online applications, it is hard to capture the true persistence of the state process without compromising classification accuracy. Narasimhan et al. (2006) proposed an online version of the Viterbi algorithm where a state is not classified until it can be done with a certain confidence. When latency is permitted, it is easier to ensure persistence. Their algorithm makes the strong assumption that uncertainty in any state label decreases with latency, which is not always the case. More recently, Garg and Pichkhadze (2019) proposed an algorithm for online decoding of Markov chain models under hard latency constraints. Meanwhile, for many practical purposes it is critical to decode states online without any latency, which is the problem we consider.

Jump models

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

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

where the loss function $\ell(y, \theta) := \|y - \theta\|_2^2$ and $\mathbb{I}$ is the indicator function, over the model parameters $\theta = \{\theta_1, \ldots, \theta_K\}$ and the state sequence $s = \{s_1, \ldots, s_T\}$. Intuitively, this objective function describes the tradeoff between fitting the data and prior assumptions about the persistence of the state sequence.

Jump penalty

The regularization parameter $\lambda \geq 0$ is a fixed-cost penalty for jumps. For λ large enough, the jump model results in a single state model. When $\lambda = 0$ the model reduces to automatically splitting the dataset in at most K states and fitting one model per state, thereby generalizing the K-means algorithm (Lloyd, 1982). Hence, the jump model allows us to infer the number of states from the data. This is in contrast to a standard HMM where the number of states is set fixed a priori, and model complexity is determined by rerunning the estimation for different values of K.

The jump penalty represents our prior knowledge about the state transition rate. It regularizes the objective function (1) by increasing the difference between states. In practice, the jump penalty needs to be selected based on the specific application in mind, for example, by cross-validation (Nystrup et al., 2020a).

Features

Exhibit 1 outlines the construction of the features we propose for learning HMMs. In addition to the time series observations themselves, they consist of the last two absolute changes and rolling means and standard deviations calculated over different windows. By computing the features for two different window lengths, $l = 6$ and $l = 14$, we obtain a total of 15 features.

The features and window lengths are closely related to the features considered by Zheng et al. (2019) and Nystrup et al. (2020b) with the important difference that they are all backward looking. The advantage of mixing forward- and backward-looking features is the inherent smoothing provided. However, in many practical applications this is not feasible. Therefore, to consider backward-looking features only is a significant improvement toward making the method useful in practical applications. We leave for future research to consider feature selection in greater depth.

Model fitting

Exhibit 2 outlines the jump model approach to estimating an HMM given a batch of time series data (Nystrup et al., 2020b). After constructing a set of standardized features the algorithm iterates between (a) finding the parameters that minimize the loss function for a given state sequence (i.e., fitting the model) and (b) finding the state sequence that minimizes the objective function (1) given the parameters (i.e., fitting the state sequence). This process is repeated until the state sequence does not change. In practice, we finish after either a maximum of ten iterations or once the value of the objective function changes by less than 10^{-6} from one iteration to the next. Usually, convergence is reached within less than five iterations. Note that although we focus on the estimation of HMMs, the jump model framework is more general and can be used to fit a much wider range of models.

Exhibit 1
Features for HMM estimation

Input: Time series $y = \{y_1, \ldots, y_t\}$ and window length l

1. Observation: y_t
2. Absolute change: $|y_t - y_{t-1}|$
3. Previous absolute change: $|y_{t-1} - y_{t-2}|$
4. Centered mean: $\text{mean}\left[y_{t-l+1}, \ldots, y_t\right]$
5. Centered standard deviation: $\text{std}\left[y_{t-l+1}, \ldots, y_t\right]$
6. Left mean: $\text{mean}\left[y_{t-l+1}, \ldots, y_{t-\frac{l}{2}}\right]$
7. Left standard deviation: $\text{std}\left[y_{t-l+1}, \ldots, y_{t-\frac{l}{2}}\right]$
8. Right mean: $\text{mean}\left[y_{t-\frac{l}{2}+1}, \ldots, y_t\right]$
9. Right standard deviation: $\text{std}\left[y_{t-\frac{l}{2}+1}, \ldots, y_t\right]$

Output: Feature set

Exhibit 2
Jump estimation of the HMM

Input: Time series $y = \{y_1, \ldots, y_T\}$, number of latent states K, jump penalty λ, and initial state sequence $s^0 = \left\{s_1^0, \ldots, s_T^0\right\}$

1. Construct a set of standardized features z from the time series y
2. Iterate for $i = 1, \ldots$
   (a) Fit model $\theta^i = \text{argmin}_\theta \sum_{t=1}^{T} \ell\left(z_t, \theta_{s_t^{i-1}}\right)$
   (b) Fit state sequence $s^i = \text{argmin}_s \left\{\sum_{t=1}^{T-1}\left[\ell\left(z_t, \theta_{s_t}^i\right) + \lambda \mathbb{I}_{s_t \neq s_{t+1}}\right] + \ell\left(z_T, \theta_{s_T}^i\right)\right\}$
3. Until $s^i = s^{i-1}$
4. Compute the transition probabilities and distributional parameters for each state

Output: HMM parameters and prediction of latent states

To improve the quality of the solution, we run the algorithm in Exhibit 2 from ten different initial state sequences generated randomly using K-means++ (Arthur and Vassilvitskii, 2007) and keep the model that achieves the lowest objective value. Based on the final estimate of the state sequence s^i, transition probabilities are computed by counting the number of transitions between states along with the distributional parameters for each state.

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

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

$$V(t, i) = \ell(z_t, \theta_i) + \min_j \left\{V(t+1, j) + \lambda \mathbb{I}_{i \neq j}\right\}, \quad t = T-1, \ldots, 1. \tag{3}$$

Then, the most likely sequence of states is given by

$$s_1 = \underset{j}{\text{argmin}}\, V(1, j), \tag{4}$$

$$s_t = \underset{j}{\text{argmin}} \left\{V(t, j) + \lambda \mathbb{I}_{s_{t-1} \neq j}\right\}, \quad t = 2, \ldots, T. \tag{5}$$

Note that if the time order of operations is reversed, (2)–(5) become the Viterbi algorithm (Viterbi, 1967).

Complexity. Finding the most likely sequence of states requires $O\left(TK^2\right)$ operations, which is the same as one forward and backward pass by the EM algorithm. While the jump estimator and EM algorithms have the same

Exhibit 3
Greedy online state classification

Input: Model parameters θ, jump penalty λ, last two observations $\{z_{t-1}, z_t\}$, and arrival cost $\mathcal{A}_{t-1}$

1. Update $\mathcal{A}_t(s_t) = \min_{s_{t-1}} \left\{ \ell(z_{t-1}, \theta_{s_{t-1}}) + \mathcal{A}_{t-1}(s_{t-1}) + \lambda \mathbb{I}_{s_{t-1} \neq s_t} \right\}$
2. Compute $\hat{s}_t = \text{argmin}_s \left\{ \ell(z_t, \theta_s) + \mathcal{A}_t(s) \right\}$

Output: Estimated state $\hat{s}_t$ and updated arrival cost $\mathcal{A}_t$

computational complexity per iteration, the jump estimator converges faster. In our simulation study, typically the jump estimator requires about five iterations, whereas the EM algorithm needs between 50 and 100 iterations to converge.

Greedy online state classification

In many practical applications it is of critical significance to estimate the model recursively in an online fashion. An advantage of our jump estimator is that it is easily adoptable to online settings. Exhibit 3 outlines how states can be inferred recursively once an initial jump model has been fit using the algorithm in Exhibit 2. It is a greedy algorithm because it estimates which state an observation belongs to without parsing historical observations. Without the fixed-cost jump penalty a greedy approach would be expected to yield less persistent state estimates.

The model parameters θ in Exhibit 3 are the same as in Exhibit 2, i.e., the state-dependent means of the feature vectors and not the HMM parameters. Importantly, our greedy online state classifier is not based on any assumptions about the conditional or sojourn-time distributions. While not explored in this paper, we remark that each time the arrival cost has been updated and the state computed, the model parameters can be updated in a straightforward way. In addition, the parameter update can easily be modified to incorporate fading memory to put greater emphasis on more recent observations (Nystrup et al., 2019). If the greedy algorithm is combined with a parameter update each time a state is estimated and the arrival cost updated, it becomes an online clustering approach.

Another advantage of our classifier is its lower memory requirements. Compared to the Viterbi algorithm, which requires storing and parsing of all historical observations, by estimating the states greedily without parsing historical (or later) observations, it only needs to store the last feature vector and arrival cost. This is a great benefit especially in real-time trading applications and in memory impoverished systems, such as IoT devices (Kumar et al., 2017, Zhu et al., 2019).

Simulation study

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

We define accuracy as the maximum classification accuracy among all possible alignments of an inferred state sequence. Specifically, accuracy is the ratio $tp/(tp + fn)$, where tp is the number of true positives and fn the number of false negatives. It is necessary to check whether a different permutation of the states yields higher accuracy due to the risk of label switching.

In our simulations, we report the balanced accuracy (BAC), which is the average accuracy per observed state, to avoid inflated performance estimates on imbalanced datasets (Brodersen et al., 2010).

Two-state HMM

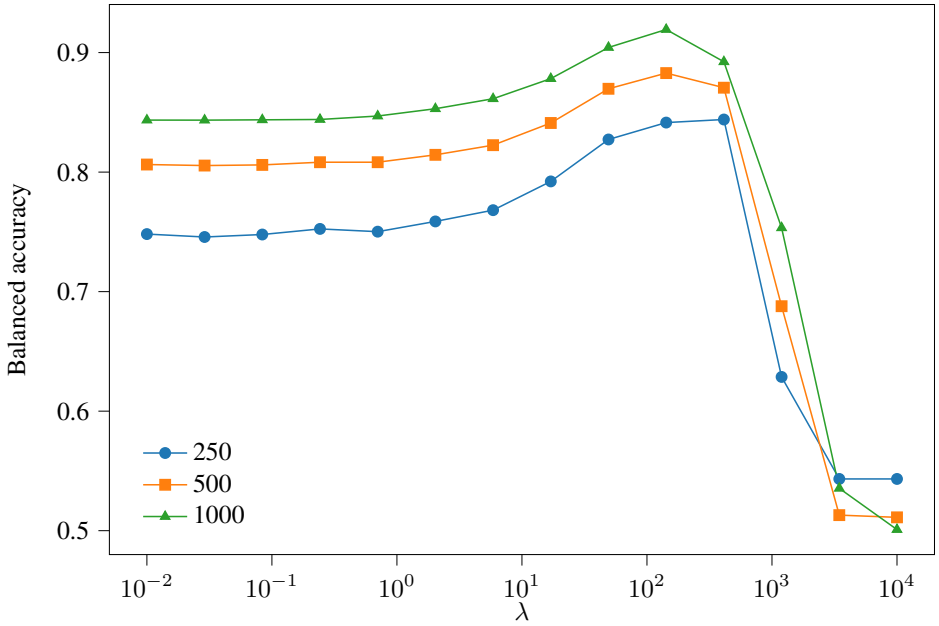
Following Zheng et al. (2019) and Nystrup et al. (2020b), we simulate data from the two-state Gaussian HMM

$$y_t | \, s_t \sim N\left(\mu_{s_t}, \sigma^2_{s_t}\right), \tag{6}$$

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

$$\begin{aligned} \mu_1 &= .0006, \qquad \mu_2 = -.0008, \\ \sigma_1 &= .0078, \qquad \sigma_2 = .0174, \\ \Gamma &= \begin{pmatrix} .9979 & .0021 \\ .0120 & .9880 \end{pmatrix}. \end{aligned} \tag{7}$$

Exhibit 4
Balanced accuracy of the jump estimator as a function of the penalty parameter for different sample lengths



These parameters are the daily equivalent of a model Hardy (2001) estimated from monthly returns on a stock market index. State one is the most persistent, has the lowest variance, and a positive mean. State two has a much higher variance and a negative mean value. This model captures the stylized behavior of many financial time series including volatility clustering and leptokurtosis (Rydén et al., 1998). The significant overlap between the conditional distributions makes it challenging to infer the unobserved states.

Selecting λ

Exhibit 4 shows the BAC of the jump estimator as a function of the penalty parameter λ for different sample lengths when applying the algorithm in Exhibit 2 to batches of data simulated from (7). Specifically, it depicts the tradeoff between accuracy and persistence. The main difference between sample lengths is the level of BAC, with longer series leading to higher BAC. When the value of the penalty parameter is too large, the BAC approaches $1/2$. When the penalty parameter is too small, the BAC converges to that of the unpenalized case, which corresponds to applying K-means clustering to the feature matrix. We find that across different sample lengths the optimal value of the penalty parameter is around $\lambda = 100$. By comparison, the optimal value is around $\lambda = 50$ when applying the greedy online classifier in Exhibit 3 based on the estimated jump model parameters.

In-sample results

Exhibit 5 shows the the mean (and standard deviation) of the estimated parameters when applying the different estimators in sample to 1000 simulated series of different lengths. The accuracy using the true parameters is based on the Viterbi path. The reported maximum likelihood estimates (MLEs) are obtained using the regularized EM algorithm (Huang et al., 2001, Chapter 9.6.1) that is implemented in the `hmmlearn` package in Python with the variance prior set to 10^{-4}. The MLE is initialized using K-means with ten different centroid seeds generated using K-means++ (Arthur and Vassilvitskii, 2007), similar to how spectral clustering and jump estimation are initialized. The accuracy of the MLEs could be improved by providing initial values that are close to the true values, but that would not be a fair comparison. Compared to the results reported by Nystrup et al. (2020b) using a mix of backward- and forward-looking features, the BACs in Exhibit 5 are only a few percentage points lower.

With a sample size of only 250 observations, the jump estimation approach outlined in Exhibit 2 with $\lambda = 100$ is better than MLE and spectral clustering at estimating nearly all parameters and states. Notably, the biggest difference is in the estimates of the transition probabilities, with MLE and spectral clustering significantly overestimating these whereas the jump estimates are close to the true values.

Exhibit 5
Parameter estimates based on 1000 simulated series of different lengths

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	ACCURACY 1	ACCURACY 2	BAC
TRUE	.0006	−.0008	.0078	.0174	.0021	.0120	.9965 (.0146)	.8752 (.2339)	.9536 (.1027)
250									
MLE	.0006 (.0021)	−.0001 (.0039)	.0074 (.0014)	.0117 (.0048)	.2884 (.2418)	.3707 (.2853)	.7747 (.2029)	**.8290** (.2351)	.7517 (.1879)
SPEC	.0006 (.0018)	−.0001 (.0035)	.0075 (.0018)	.0118 (.0046)	.0997 (.0535)	.1612 (.0589)	.7175 (.1874)	.7589 (.1890)	.6924 (.1526)
JUMP	.0006 (.0009)	−.0001 (.0031)	**.0079** (.0014)	**.0120** (.0049)	**.0055** (.0062)	**.0242** (.0277)	**.8611** (.1460)	.8263 (.2398)	**.8303** (.1457)
500									
MLE	.0007 (.0012)	−.0003 (.0023)	.0075 (.0009)	.0139 (.0046)	.1805 (.2314)	.2289 (.2612)	.8648 (.1892)	**.8647** (.2074)	.8293 (.1852)
SPEC	.0007 (.0014)	−**.0005** (.0031)	.0075 (.0012)	.0137 (.0045)	.0773 (.0559)	.1463 (.0604)	.7913 (.1867)	.7728 (.1895)	.7562 (.1549)
JUMP	**.0006** (.0005)	−.0003 (.0028)	**.0079** (.0005)	**.0143** (.0046)	**.0034** (.0024)	**.0204** (.0206)	**.9198** (.1239)	.8458 (.2095)	**.8736** (.1382)
1000									
MLE	**.0006** (.0006)	−.0005 (.0017)	.0076 (.0004)	.0154 (.0038)	.0962 (.1799)	.1382 (.2140)	.9365 (.1327)	**.8846** (.1776)	.8961 (.1447)
SPEC	.0008 (.0009)	−.0010 (.0024)	.0076 (.0008)	.0149 (.0039)	.0534 (.0463)	.1325 (.0568)	.8521 (.1599)	.7936 (.1630)	.8140 (.1248)
JUMP	**.0006** (.0003)	−**.0007** (.0023)	**.0078** (.0003)	**.0160** (.0034)	**.0025** (.0015)	**.0188** (.0163)	**.9669** (.0727)	.8727 (.1509)	**.9173** (.0941)

Notes: The reported values are the mean (and standard deviation) of the estimated parameters. Bold entries identify the best estimate between MLE, spectral, and jump estimation.

Exhibit 6
Parameter estimates based on 1000 series of length 500 simulated from conditional t_5-distributions

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	ACCURACY 1	ACCURACY 2	BAC
TRUE	.0006	−.0008	.0078	.0174	.0021	.0120	.9869 (.0228)	.8238 (.2752)	.9291 (.1231)
MLE	**.0006** (.0005)	−.0011 (.0160)	.0066 (.0021)	.0166 (.0047)	.1175 (.1341)	.4149 (.3488)	.9260 (.1102)	.6936 (.2931)	.8368 (.1456)
SPEC	.0007 (.0016)	−.0005 (.0040)	**.0076** (.0031)	.0147 (.0059)	.0565 (.0398)	.1467 (.0646)	.8023 (.1634)	.6887 (.2424)	.7481 (.1351)
JUMP	**.0006** (.0016)	−**.0006** (.0054)	**.0080** (.0037)	**.0171** (.0088)	**.0043** (.0106)	**.0386** (.0427)	**.9374** (.1216)	**.7432** (.2941)	**.8587** (.1497)

Notes: The reported values are the mean (and standard deviation) of the estimated parameters. Bold entries identify the best estimate between MLE, spectral, and jump estimation.

With a sample size of 500 observations, the BAC of the jump estimate is still higher than that of the MLE, with spectral clustering far behind. Although the MLE of the transition probabilities have improved, they are still far from the true values.

Even with a sample size of 1000 observations, neither spectral clustering nor MLE are able to capture the true persistence of the underlying state process. The BAC of MLE is slightly lower than that of the jump estimator, and both methods are fairly close to the accuracy of the Viterbi path based on the true parameter values.

Of course, if we kept increasing the sample size, the performance of MLE would keep improving. But it is not realistic for many practical purposes to have more than 1000 daily observations that are representative of current conditions available for estimation.

Misspecified conditional distributions

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

Exhibit 6 shows the result when simulating 1000 series of 500 observations from Student's t distributions. The BACs are about the same as in Exhibit 5. Sampling from conditional distributions with heavier tails significantly inflates the transition probability estimates by MLE, leading to an estimate of 41% compared to the true value of 1% probability of jumping from state two to one. Both the spectral clustering and jump estimators are far more robust to the misspecified conditional distributions.

Misspecified sojourn-time distributions

In order to compare the different estimation methods when sojourn-time distributions are misspecified, we repeat the simulation study, only this time we sample the sojourn times from negative binomial distributions instead of geometric distributions. The negative binomial distributions are calibrated to yield the same expected sojourn duration as before by adjusting the probability parameters. We fix the shape parameter to $n_1 = 0.1$ in state one and $n_2 = 0.06$ in state

Exhibit 7
Parameter estimates based on 1000 series of length 500 simulated from negative binomial sojourn-time distributions

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	ACCURACY 1	ACCURACY 2	BAC
TRUE	.0006	−.0008	.0078	.0174	.0021*	.0120*	.9739 (.1369)	.8329 (.3272)	.9375 (.1449)
MLE	.0003 (.0027)	.0001 (.0032)	**.0084** (.0031)	.0123 (.0045)	.3085 (.2417)	.3449 (.2534)	.7682 (.2165)	.7880 (.2551)	.7396 (.1998)
SPEC	**.0006** (.0023)	−**.0005** (.0030)	.0087 (.0038)	.0126 (.0044)	.1072 (.0579)	.1517 (.0558)	.7191 (.2264)	.7437 (.2185)	.6943 (.1756)
JUMP	.0005 (.0018)	−.0002 (.0025)	.0088 (.0031)	**.0129** (.0047)	**.0053** (.0088)	**.0189** (.0232)	**.8663** (.1858)	**.8434** (.2510)	**.8420** (.1648)

Notes: The reported values are the mean (and standard deviation) of the estimated parameters. Bold entries identify the best estimate between MLE, spectral, and jump estimation.

Exhibit 8
Parameters estimated online out of sample based on 1000 simulated series, when different sample lengths were available for in-sample estimation

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	ACCURACY 1	ACCURACY 2	BAC
TRUE	.0006	−.0008	.0078	.0174	.0021	.0120	.9695 (.0743)	.8014 (.2767)	.9367 (.1146)
250									
MLE	.0005 (.0010)	.0003 (.0018)	.0087 (.0019)	.0107 (.0038)	.2455 (.2353)	.4910 (.3031)	.7007 (.2327)	**.7099** (.2408)	.7175 (.1858)
JUMP	.0005 (.0011)	**.0002** (.0020)	**.0084** (.0014)	**.0114** (.0043)	**.0153** (.0358)	**.0676** (.1001)	**.8420** (.1797)	.6635 (.3457)	**.8020** (.1777)
500									
MLE	.0005 (.0011)	.0001 (.0019)	.0085 (.0017)	.0119 (.0042)	.1491 (.2123)	.3852 (.3226)	.7923 (.2300)	**.7779** (.2477)	.8060 (.1936)
JUMP	**.0006** (.0008)	−**.0001** (.0025)	**.0084** (.0016)	**.0128** (.0044)	**.0109** (.0178)	**.0606** (.0906)	**.8976** (.1627)	.7120 (.3209)	**.8586** (.1684)
1000									
MLE	.0006 (.0007)	.0000 (.0024)	.0083 (.0012)	.0131 (.0044)	.0687 (.1391)	.2743 (.3123)	.8563 (.2128)	**.8408** (.2307)	.8678 (.1633)
JUMP	.0006 (.0006)	−**.0001** (.0026)	**.0082** (.0011)	**.0141** (.0044)	**.0072** (.0134)	**.0611** (.1130)	**.9487** (.0992)	.7216 (.2995)	**.8953** (.1432)

Notes: The out-of-sample testing is done on samples of length 250. The reported values are the mean (and standard deviation) of the estimated parameters. Bold entries identify the best estimate between the MLE-based Viterbi states and the greedy jump states.

two based on the values Bulla and Bulla (2006) reported from fitting similar hidden semi-Markov models to daily stock returns. This leads to an increased number of very short sojourns as well as an increased number of prolonged sojourns.

Exhibit 7 shows the result when sampling from misspecified sojourn-time distributions. The negative binomial distribution cannot be summarized by a single transition probability parameter. However, the inverse of the "true" and estimated transition probabilities correspond to the average sojourn duration. More so than when sampling from misspecified conditional distributions, the accuracy of the MLEs deteriorates significantly compared to Exhibit 5, whereas the jump estimator is remarkably robust. The jump estimator maintains a high accuracy for both states as well as for the sojourn duration.

Out-of-sample results

Next, we consider an out-of-sample application where observations are processed sequentially one-by-one. We start with the Gaussian models that were fit above based on 1000 series of different lengths simulated using (7). We apply these models sequentially to subsequent samples of 250 observations without reestimating any parameters. Because spectral clustering was not able to contend with MLE and the jump estimator in sample, we choose to focus on the latter two in the out-of-sample study.

Online state classification without latency is a significantly harder task than in-sample state inference on a batch of data where filtering algorithms can be applied to smooth the estimated state sequence. Based on the in-sample MLE we apply the Viterbi algorithm sequentially to the out-of-sample test data, every time outputting the state corresponding to the last observation. For the jump models, we apply the algorithm in Exhibit 3 with $\lambda = 50$ to greedily infer the states underlying the out-of-sample test data. It is natural to expect that a greedy approach will lead to lower classification accuracy compared to an approach that parses historical observations before classifying each new observation; but that is not the case in our study.

Exhibit 8 shows the result when inferring the hidden states sequentially on 1000 simulated series of length 250 based on the models estimated in sample from samples of different length. The reported transition probabilities are based on counting the number of transitions while the means and variances are estimated conditional on the classified state sequences. Compared to Exhibit 5, the BACs are 2%- to 4%-points lower. Interestingly, both estimators have a much harder time identifying state two out of sample compared to in sample.

Exhibit 9
Parameters estimated using the greedy online classifier out of sample based on 1000 simulated series when observations sampled at higher frequencies were available for estimation of the volatility features

	μ_1	μ_2	σ_1	σ_2	γ_{12}	γ_{21}	ACCURACY 1	ACCURACY 2	BAC
TRUE	.0006	−.0008	.0078	.0174	.0021	.0120	.9695 (.0743)	.8014 (.2767)	.9367 (.1146)
2	.0006 (.0007)	.0000 (.0025)	.0082 (.0012)	.0128 (.0045)	.0106 (.0361)	.0555 (.1004)	.9020 (.1567)	.7508 (.3327)	.8676 (.1693)
5	.0006 (.0007)	−.0001 (.0025)	.0081 (.0009)	.0130 (.0044)	.0088 (.0195)	.0459 (.0864)	.9029 (.1595)	.7760 (.3315)	.8770 (.1704)
10	.0006 (.0007)	.0000 (.0022)	.0080 (.0010)	.0131 (.0044)	.0082 (.0173)	.0528 (.1148)	.9068 (.1587)	.7991 (.3131)	.8856 (.1593)

Notes: Each series consists of 500 observations for in-sample estimation and 250 observations for out-of-sample testing. The reported values are the mean (and standard deviation) of the estimated parameters.

Similar to the in-sample study, the most significant difference between the MLE-based and the greedy state classifications is the estimated transition frequencies. In the out-of-sample test, the implied transition probabilities based on the MLE state sequences are significantly worse. Although the greedy estimates of the transition probabilities are higher than the in-sample jump estimates, they are still much closer to the true values.

Intraday data

The results presented in Exhibit 8 made us wonder what could be done to improve the greedy classifier's ability to identify observations belonging to the high-volatility state. Besides its robustness and ability to estimate persistent state processes, the biggest advantage of the jump estimator as compared to MLE is its potential for easily extending the feature space.

As a simple illustration, we explore how accuracy can be improved when observations sampled at a higher frequency than daily are available for estimation of the standard deviation features in Exhibit 1. In financial applications, it is common to use intraday data to improve estimates of daily volatility (Barndorff-Nielsen and Shephard, 2002).

Exhibit 9 shows the result when observations sampled at a higher frequency are available for estimation of the standard deviation features. It is apparent that sampling at frequencies of two, five, and ten observations per day significantly improves the greedy classifier's accuracy for state two. Compared to Exhibit 8, when two observations per day are available for feature estimation, the improvement in BAC is about 1%-point. With five and ten observations per day available for feature estimation, the improvements in BAC are 2%- and 3%-points, respectively. By sampling ten times per day, the BAC of the greedy classifier is 1%-point higher than the in-sample BAC in Exhibit 5. This result is very encouraging considering how much harder online out-of-sample state classification is. In addition, we note that by increasing the sampling frequency the estimates of the transition probabilities improve in comparison to Exhibit 8.

Application to S&P 500

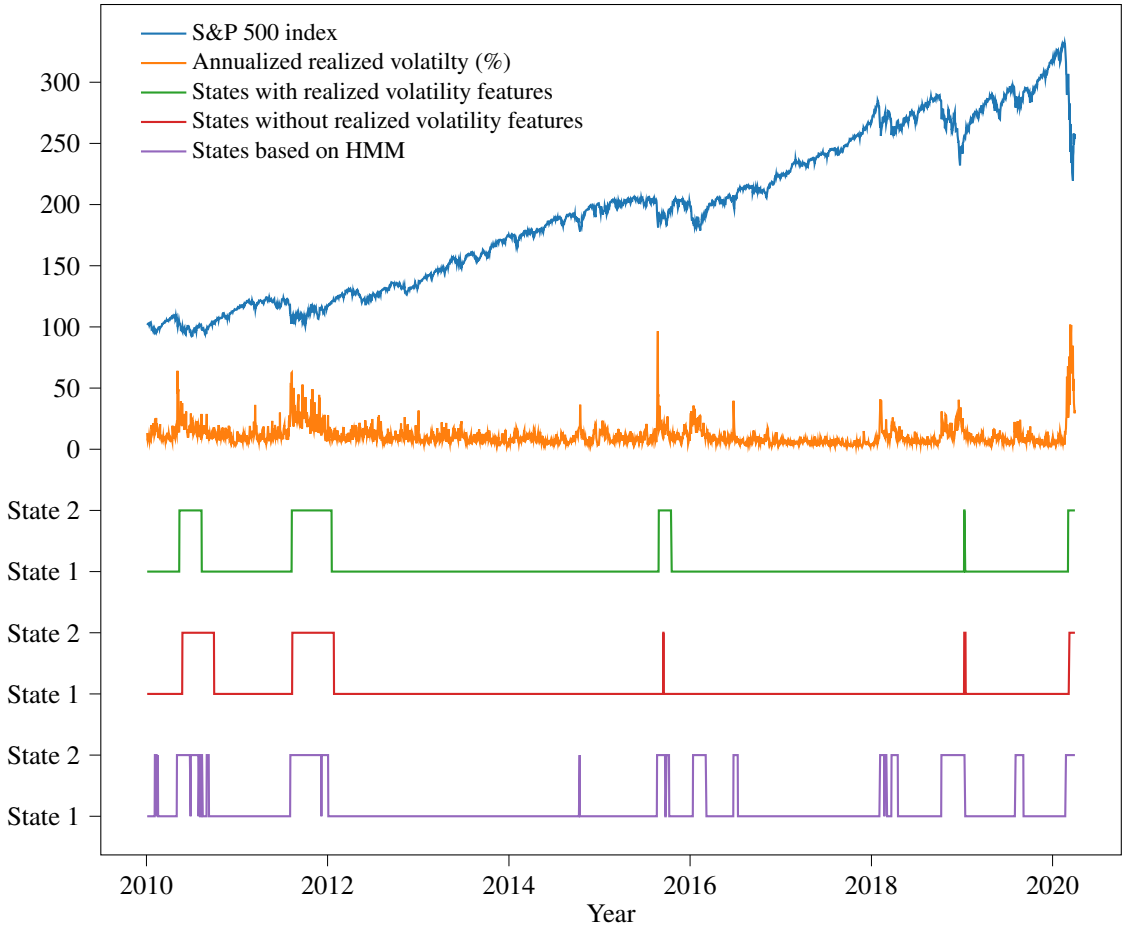
Finally, we consider an application to the S&P 500 index. We consider daily log-returns along with five-minute sub-sampled realized volatility data covering the period from January 2000 through March 2020.[1] First, we use the data from January 2000 to January 2005 for fitting a jump model with $\lambda = 1000$ using the algorithm in Exhibit 2. Second, we use the data from January 2005 to January 2010 for selecting a value $\lambda = 500$ to use in the greedy online classifier in Exhibit 3. Third, we use the greedy online classifier to estimate states for the period from January 2010 through March 2020.

Exhibit 10 shows the S&P 500 index and its annualized realized volatility in percent (five-minute sub-sampled) on the test period from January 2010 through March 2020 along with three different state sequences. The first two state sequences are estimated using the greedy online classifier in Exhibit 3 based on jump models with and without the realized volatility features. The third state sequence is estimated by applying the Viterbi algorithm sequentially to the test data, based on an HMM fitted to the log-returns from January 2000 to January 2010 using maximum likelihood estimation.

Adding two extra features, which are moving averages with window lengths $l = 6$ and $l = 14$ of the five-minute sub-sampled realized volatility observations, leads to improved state estimates. In particular, the greedy classifier is able to detect state changes slightly faster based on the realized volatility features. Although the differences are measured

[1]The five-minute sub-sampled realized volatility data for the S&P 500 index was downloaded from `https://realized.oxford-man.ox.ac.uk/data/download`.

Exhibit 10
Comparison of state sequences estimated based on the S&P 500 index and its realized volatility



Note: The legends are sorted according to the order of the time series.

in days and not months, they can still make a substantial difference, because they happen during the most turbulent times.

Comparing the two state sequences that are based on jump models to that based on an HMM, the differences in persistence are striking. The HMM detects changes equally quickly as the jump model with the realized volatility features; however, this comes at the cost of many spurious and short-lived state changes. If the estimated states were used as the basis of a regime-based asset allocation strategy, then the lower persistence would lead to a substantially higher turnover, which could easily make the difference between whether the strategy is profitable or not (Nystrup et al., 2015a, 2017a, 2018). The results from this application suggest that using data sampled at higher frequencies can help improve the classification of persistent states in financial time series. Determination of the optimal sampling frequency is outside the scope of this article (Ait-Sahalia et al., 2005).

Conclusion

We proposed a greedy online classifier that contemporaneously determines which hidden state a new observation belongs to without the need to parse historical observations and without compromising persistence. We believe this classifier will be useful in various practical applications where it is important to classify time series data without any latency while maintaining persistence in the identified states. Through a series of simulations, we showed that 1) in most settings the new classifier remarkably obtains a higher accuracy than the correctly specified maximum likelihood estimator and 2) the new classifier is more robust to misspecification, yielding state sequences that are significantly

more persistent both in and out of sample. Additionally, we demonstrated how classification accuracy can be further improved by including features that are based on data sampled at higher frequencies.

The MLE required at least a thousand observations in order to compete with the proposed jump estimator based on backward-looking temporal features and penalizing jumps, a strong argument in favor of the jump estimator. Furthermore, we demonstrated that the jump estimates of the transition probabilities were significantly more precise. In the sequential out-of-sample test, where it was harder to obtain persistent state classifications because of the inability to apply smoothing, the differences between the states obtained from MLE and the jump model were even more pronounced.

The robustness of the jump estimator to initialization, misspecification, and data scarcity should make it the preferred choice for practical applications where, by definition, all models are wrong. With the same computational complexity per iteration as the EM algorithm, it is a significant advantage of the jump estimator that it needs 10 to 20 times fewer iterations to converge. Additionally, the computational advantage of the jump estimator will be particularly useful in high-frequency applications. As demonstrated in the simulation study, higher sampling rates can lead to accuracy gains.

Future work. In closing, we comment on some directions for future work. In many situations the researcher has access to application specific exogenous data which may improve state classification if added to the feature space. For example, for time series of security returns and changes in macroeconomic variables we hypothesize that news- and sentiment-based features could be useful. In these situations, it is natural to consider approaches for determining the appropriate features to include. In a future paper, we will explore feature selection techniques in the jump estimation framework.

We anticipate that a generalization of the framework to multivariate HMMs is straightforward, using the features in Exhibit 1 as well as features that exploit the dependence between variables. We believe that the robustness of the jump framework will be beneficial in clustering high-dimensional time series with large feature sets.

In future work we plan to leverage the increased persistence of the jump model in regime-based asset allocation strategies.

Acknowledgments

This work was supported by Vergstiftelsen and the Centre for IT-Intelligent Energy Systems (CITIES) project funded in part by Innovation Fund Denmark under Grant No. 1305-00027B.

References

Ait-Sahalia, Y., P. A. Mykland, and L. Zhang. "How often to sample a continuous-time process in the presence of market microstructure noise." *Review of Financial Studies*, vol. 18, no. 2 (2005), pp. 351–416.

Andersson, S., T. Rydén, and R. Johansson. "Linear optimal prediction and innovations representations of hidden Markov models." *Stochastic Processes and their Applications*, vol. 108, no. 1 (2003), pp. 131–149.

Arthur, D. and S. Vassilvitskii. "k-means++: The advantages of careful seeding." In *Proceedings of the eighteenth annual ACM–SIAM symposium on Discrete algorithms* (2007), pp. 1027–1035.

Barndorff-Nielsen, O. E. and N. Shephard. "Econometric analysis of realized volatility and its use in estimating stochastic volatility models." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 64, no. 2 (2002), pp. 253–280.

Baum, L. E., T. Petrie, G. Soules, and N. Weiss. "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains." *Annals of Mathematical Statistics*, vol. 41, no. 1 (1970), pp. 164–171.

Beal, M. J., Z. Ghahramani, and C. E. Rasmussen. "The infinite hidden Markov model." In *Advances in Neural Information Processing Systems* (2002), pp. 577–584.

Bellman, R. E. *Dynamic programming*. Princeton University Press: Princeton (1957).

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

Bhardwaj, S., V. Sharma, S. Srivastava, O. Sastry, B. Bandyopadhyay, S. Chandel, and J. Gupta. "Estimation of solar radiation using a combination of hidden Markov model and generalized fuzzy model." *Solar Energy*, vol. 93 (2013), pp. 43–54.

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

Bulla, J. "Hidden Markov models with t components. Increased persistence and other aspects." *Quantitative Finance*, vol. 11, no. 3 (2011), pp. 459–475.

Bulla, J. and I. Bulla. "Stylized facts of financial time series and hidden semi-Markov models." *Computational Statistics & Data Analysis*, vol. 51, no. 4 (2006), pp. 2192–2209.

Cont, R. "Empirical properties of asset returns: stylized facts and statistical issues." *Quantitative Finance*, vol. 1, no. 2 (2001), pp. 223–236.

Dempster, A. P., N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 39, no. 1 (1977), pp. 1–38.

Dias, J. G. and S. B. Ramos. "Dynamic clustering of energy markets: An extended hidden Markov approach." *Expert Systems with Applications*, vol. 41, no. 17 (2014), pp. 7722–7729.

Fiecas, M., J. Franke, R. von Sachs, and J. T. Kamgaing. "Shrinkage estimation for multivariate hidden Markov models." *Journal of the American Statistical Association*, vol. 112, no. 517 (2017), pp. 424–435.

Fox, E. B., E. B. Sudderth, M. I. Jordan, and A. S. Willsky. "A sticky HDP-HMM with application to speaker diarization." *Annals of Applied Statistics*, vol. 5, no. 2A (2011), pp. 1020–1056.

Gales, M. and S. Young. "The application of hidden Markov models in speech recognition." *Foundations and Trends in Signal Processing*, vol. 1, no. 3 (2008), pp. 195–304.

Garg, V. and T. Pichkhadze. "Online Markov decoding: Lower bounds and near-optimal approximation algorithms." In *Advances in Neural Information Processing Systems* (2019), pp. 5681–5691.

Ghahramani, Z. and M. I. Jordan. "Factorial hidden Markov models." *Machine Learning*, vol. 29, no. 2–3 (1997), pp. 245–273.

Hallac, D., P. Nystrup, and S. Boyd. "Greedy Gaussian segmentation of multivariate time series." *Advances in Data Analysis and Classification*, vol. 13, no. 3 (2019), pp. 727–751.

Hardy, M. R. "A regime-switching model of long-term stock returns." *North American Actuarial Journal*, vol. 5, no. 2 (2001), pp. 41–53.

Huang, X., A. Acero, and H. W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*. Prentice–Hall: New Jersey (2001).

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

Kim, S. J., K. Koh, S. Boyd, and D. Gorinevsky. "ℓ_1 trend filtering." *SIAM Review*, vol. 51, no. 2 (2009), pp. 339–360.

Kumar, A., S. Goyal, and M. Varma. "Resource-efficient machine learning in 2 kb ram for the internet of things." In *Proceedings of the 34th International Conference on Machine Learning* (2017), pp. 1935–1944.

Langrock, R. and W. Zucchini. "Hidden Markov models with arbitrary state dwell-time distributions." *Computational Statistics & Data Analysis*, vol. 55, no. 1 (2011), pp. 715–724.

Lloyd, S. P. "Least squares quantization in PCM." *IEEE Transactions on Information Theory*, vol. 28, no. 2 (1982), pp. 129–137.

Narasimhan, M., P. Viola, and M. Shilman. "Online decoding of Markov models under latency constraints." In *Proceedings of the 23rd International Conference on Machine Learning* (2006), pp. 657–664.

Nefian, A. V. and M. H. Hayes. "Hidden Markov models for face recognition." In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5 (1998), pp. 2721–2724.

Ng, A. Y., M. I. Jordan, and Y. Weiss. "On spectral clustering: Analysis and an algorithm." In *Advances in Neural Information Processing Systems* (2002), pp. 849–856.

Nystrup, P., S. Boyd, E. Lindström, and H. Madsen. "Multi-period portfolio selection with drawdown control." *Annals of Operations Research*, vol. 282, no. 1–2 (2019), pp. 245–271.

Nystrup, P., B. W. Hansen, H. O. Larsen, H. Madsen, and E. Lindström. "Dynamic allocation or diversification: A regime-based approach to multiple assets." *Journal of Portfolio Management*, vol. 44, no. 2 (2017a), pp. 62–73.

Nystrup, P., B. W. Hansen, H. Madsen, and E. Lindström. "Regime-based versus static asset allocation: Letting the data speak." *Journal of Portfolio Management*, vol. 42, no. 1 (2015a), pp. 103–109.

———. "Detecting change points in VIX and S&P 500: A new approach to dynamic asset allocation." *Journal of Asset Management*, vol. 17, no. 5 (2016), pp. 361–374.

Nystrup, P., E. Lindström, and H. Madsen. "Hyperparameter optimization for portfolio selection." *Journal of Financial Data Science*, vol. 2, no. 3 (2020a).

———. "Learning hidden Markov models with persistent states by penalizing jumps." *Expert Systems with Applications*, vol. 150 (2020b), p. 113307.

Nystrup, P., H. Madsen, and E. Lindström. "Stylised facts of financial time series and hidden Markov models in continuous time." *Quantitative Finance*, vol. 15, no. 9 (2015b), pp. 1531–1541.

———. "Long memory of financial time series and hidden Markov models with time-varying parameters." *Journal of Forecasting*, vol. 36, no. 8 (2017b), pp. 989–1002.

———. "Dynamic portfolio optimization across hidden market regimes." *Quantitative Finance*, vol. 18, no. 1 (2018), pp. 83–95.

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

Pinson, P., L. Christensen, H. Madsen, P. Sørensen, M. Donovan, and L. Jensen. "Regime-switching modelling of the fluctuations of offshore wind generation." *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 96, no. 12 (2008), pp. 2327–2347.

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

Rousseeuw, K., Émilie Poisson Caillault, A. Lefebvre, and D. Hamad. "Hybrid hidden Markov model for marine environment monitoring." *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 8, no. 1 (2015), pp. 204–213.

Rydén, T. "EM versus Markov chain Monte Carlo for estimation of hidden Markov models: A computational perspective." *Bayesian Analysis*, vol. 3, no. 4 (2008), pp. 659–688.

Rydén, T., T. Teräsvirta, and S. Åsbrink. "Stylized facts of daily return series and the hidden Markov model." *Journal of Applied Econometrics*, vol. 13, no. 3 (1998), pp. 217–244.

Samé, A., F. Chamroukhi, G. Govaert, and P. Aknin. "Model-based clustering and segmentation of time series with changes in regime." *Advances in Data Analysis and Classification*, vol. 5, no. 4 (2011), pp. 301–321.

Shamshad, A., M. A. Bawadi, W. M. A. W. Hussin, T. A. Majid, and S. A. M. Sanusi. "First and second order Markov chain models for synthetic generation of wind speed time series." *Energy*, vol. 30, no. 5 (2005), pp. 693–708.

Verbeek, J. J., N. Vlassis, and B. Kröse. "Efficient greedy learning of Gaussian mixture models." *Neural Computation*, vol. 15, no. 2 (2003), pp. 469–485.

Viterbi, A. J. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm." *IEEE Transactions on Information Theory*, vol. 13, no. 2 (1967), pp. 260–269.

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

Zhu, P., D. A. E. Acar, N. Feng, P. Jain, and V. Saligrama. "Cost aware inference for IoT devices." In *The 22nd International Conference on Artificial Intelligence and Statistics* (2019), pp. 2770–2779.