Jump Estimation of Hidden Markov Models with Time-Varying Transition Probabilities

Henning Tansjö

Bachelor's thesis
2020:K9

Lund University

Faculty of Science
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Lund University

Mathematical Statistics: Bachelor's Degree Project

MASK11

Jump Estimation of Hidden Markov Models with Time-Varying Transition Probabilities

Author:
Henning Tansjö

Supervisor:
Prof. Erik Lindström

June 2020

Abstract

The Hidden Markov model is applicable to a wide variety of fields. Applied to financial time series, its assumed underlying state sequence can reflect the time series' tendency to behave differently over different periods of time. In many situations, models could be improved by including exogenous data. However, that may in some cases be inappropriate in practice as the model could get too mathematically complex to learn or require too strong assumptions. The jump estimator for learning Hidden Markov models by clustering temporal features is very flexible in that regard. In this thesis we conduct a simulation study to show that, assuming time-varying transition probabilities depending on exogenous variables, the jump estimator's prediction accuracy of latent states can be significantly improved as its feature space is extended with the relevant exogenous data. To facilitate the simulation study, we use an EM-algorithm to estimate Hidden Markov models applied to the S&P 500 index with transition probabilities depending on exogenous variables. Four variables are considered in a forward selection scheme resulting in the CBOE Volatility Index being deemed the most important exogenous variable in this setting. For practical purposes, our results indicate in particular that when applying the jump estimator to the S&P 500, including features based on the volatility index improves its ability to segment the S&P 500 into periods with bullish and bearish market conditions.

Populärvetenskaplig sammanfattning

En *Dold Markovmodell* (HMM) kan användas för modellera tidsserier på ett sätt som beskriver olika beteenden under olika perioder; till exempel kan den beskriva finansiella tidsseriers benägenhet att plötsligt ändra riktning då marknadssentimentet förändras, exempelvis på grund av en pandemi. Givet en tidsserie, tänker man sig att den antar olika tillstånd som ej går att observera men som reflekteras i hur variabeln i tidsserien är slumpartat fördelad. Systemet kan ses som en underliggande ”dold” process som utvecklas stegvist i tid och slumpmässigt hoppar mellan olika tillstånd samt emitterar en observation vid varje tidpunkt. Denna process kallas för Markovkedja och definieras av att sannolikheterna att hamna i olika tillstånd i framtiden endast beror på nuet.

När man modellerar en tidsserie med en HMM vill man, baserat på tidigare observationer, (1) skatta parametrar som har att göra med hur den observerbara variabeln fördelas under olika tillstånd samt sannolikheterna som beskriver övergången mellan tillstånden och, (2) försöka bestämma, ”prediktera”, vilka tillstånd som har genererat varje observation i tidsserien. Traditionellt sett har detta gjorts med den så kallade *Maximum Likelihood-metoden*. Denna uppsats bygger på en nyligen föreslagen klustringsmetod för att lösa både parameterskattning och prediktion samtidigt. Den utnyttjar olikheter i fördelningen av olika observationer som genererats under olika tillstånd för att ordna dem i grupper som representerar de möjliga tillstånden. Till skillnad från Maximum Likelihood-metoden behöver den inte göra lika starka antaganden om systemet som modelleras, vilket minskar risken för att den blir vilseledande. Vidare kan man med fördel kontrollera hur stabila de predikterade tillstånden ska vara genom en regulariseringsparameter. Detta kan till exempel leda till lägre transaktionskostnader om en handelsstrategi baseras på modellen.

I en ”vanlig” HMM är övergångssannoliketerna i Markovkedjan konstanta. Denna kan modifieras genom att låta sannolikheterna bero på exogena variabler som skulle kunna ha en påverkan på den modellerade tidsserien. Detta skulle å ena sidan kunna förbättra modellen, men å andra sidan kan den bli svårare att arbeta med. Den ovan nämnda klustermetoden för HMMer, *Jump-metoden*, är däremot mycket flexibel i det avseendet och kan enkelt utvidgas för att ta hänsyn till exogen data. I denna uppsats visar vi genom en simuleringsstudie att, under antagandet att övergångssannoliketerna beror på exogena variabler, kan Jump-metodens prediktiva förmåga förbättras då den, utöver observationer från den modellerade (endogena) tidsserien, även klustrar de relevanta exogena variablerna. Simulerade tidsserier tillåter en jämförelse av Jump-metodens predikterade tillstånd vid varje tidpunkt med de ”verkliga” (som annars är okända) och därmed går det att bedöma metodens träffsäkerhet. Vi simulerar tidsserierna enligt HMMer av senast nämnda sort ämnade att modellera Standard & Poor’s 500 index med exogena variabler valda så att modellerna passar observerad data väl med statistiska mått. Variabelurvalet tyder på att, framför allt, CBOE Volatility Index med fördel kan inkluderas som exogen variabel. Detta samt de positiva resultaten från simuleringsstudien verkar vara användbara i praktiken; då Jump-metoden även klustrar CBOE Volatility Index, tyder våra resultat på att dess förmåga att identifiera perioder i Standard & Poor’s 500 med, i genomsnitt, antingen positiv eller negativ avkastning förbättras.

Acknowledgements

I would like to thank my supervisor Prof. Erik Lindström who introduced me to this project and gave me guidance and support throughout the process. I would also like to express my appreciation to Dr. Peter Nystrup for his useful comments and our interesting discussions. He helped me get started and his interest in my results has been encouraging.

Contents

List of Tables

List of Figures

List of Algorithms

1 Introduction

The Hidden Markov model (HMM) is applicable to a wide range of fields including computational biology; speech recognition; and finance. Modeling a system where one can observe a sequence of variables, e.g. a time series, it may be appropriate to assume that different observations in that sequence was generated under different conditions. For example, in a time series of financial returns, one reasonable assumption is that the market can be upwards trending (bullish) or downwards trending (bearish) and that the distribution of the returns are determined by these states. This situation could be modeled with a HMM. It assumes that the distribution of an observable variable is completely determined by an associated unobservable (hidden) state and that the probabilities of transitioning to each state at the next point in time is only dependent on the current state.

When modeling a system with a HMM it is of interest to estimate the parameters of the model; that is the parameters related to the distributions of the outputted variable under each state and the transition probabilities. With a sequence of observed outputted variables and the estimated HMM, one would also be interested in predicting the corresponding most likely sequence of hidden states. Traditionally, in the case with a finite amount of states and discrete time, the parameters are estimated by Maximum likelihood estimation (MLE) after which the most likely sequence of states is predicted using the Viterbi algorithm (Viterbi, 1967).

When building a model, one has to make assumptions to simplify the truth sufficiently much for it to be explained by the model. The model will be wrong but possibly useful. Consider a map which is a very simplified (but useful) model of some part of earth's surface. If the assumptions of the model are too far off and the model is too dependent on them being correct; it may however break down and give a deceptive picture of reality. It is therefore desirable to develop models with relaxed assumptions while still being sufficiently informative. MLE requires strong assumptions about the HMM and is therefore not very robust to misspecification of, for example, the conditional distributions given each state.

Zheng, Li and Xu (2019) proposed to apply spectral clustering to a set of features derived from the observed time series of the outputted variable (when continuous) in order to learn the HMM. It relaxes the assumptions of MLE about the conditional distributions. However, their method does not consider the temporal order of the observations in the time series. Therefore, relevant information such as autocorrelation is not accounted for. In particular for financial time series of returns; empirical studies show patterns such as volatility tending to cluster over several days (Cont, 2001) suggesting persistent states.

To address this, Nystrup, Lindström and Madsen (2020b) combined the jump framework of Bemporad, Breschi, Piga and Boyd (2019) with the temporal features used by Zheng et al. (2019) in order to learn HMMs. Their proposed way of learning HMMs, referred to as the *jump estimator*, is a clustering method with a regularisation parameter penalising jumps between states. This allows advantageous control over the transition rates in the predicted state sequence. In their simulation study, it performed favourably over both MLE and spectral clustering in several regards (Nystrup et al. 2020b). Like spectral clustering, their method make less assumptions than the MLE about the observed time series; it does not require that the observed data follows any specific distribution. Furthermore, being a clustering method, one

can effortlessly extend the clustered feature space with exogenous variables assumed to contain useful information.

This thesis shall treat that prospect. We aim to investigate if the jump estimator may be further improved by adding relevant exogenous features to the feature space. More specifically, we shall explore the potential the jump estimator applied to the Standard & Poor's 500 (S&P 500) index assuming a HMM with transition probabilities depending on exogenous variables. We aim to find exogenous variables that are appropriate to include in such a model, and compare the performance of the jump estimator before and after adding those variables to the "vanilla" feature space used by Nystrup et al. (2020a).

In Section 2, we describe and estimate a HMM explaining the S&P 500 log-returns with exogenous variables determining the transition probabilities. It is the model proposed by Diebold, Lee and Weinbach (1994) and is fit using their expectaion-maximisation (EM) algorithm. We consider four variables to include and perform forward selection to find the ones which result in the best model. This is to be done in two approaches differing in how the data is preprocessed; resulting in two HMMs.

The jump estimator of Nystrup et al. (2020b) for learning HMMs is explained in greater detailed in Section 3. After this, we do a simulation study in Section 4 based on time series generated according to the HMMs from Section 2. This allows to assess the performance of the jump estimator using different feature spaces and inputs. Before concluding and commenting on further work in Section 6, we apply it to real data in Section 5 in order to demonstrate the practical potential of what is learned throughout the thesis.

As this thesis can be divided into two parts with results obtained in Sections 2 and 4, we end each of those sections with a discussion. Firstly however, we go through some theoretical preliminaries beyond basic mathematical statistics necessary to understand the contents in this text.

1.1 Preliminaries

1.1.1 Basic concepts in Finance

A stock index measures the overall development of a stock market. The price of a stock index is derived from the prices of some selected stocks representing the market (for example by a weighted arithmetic mean).

The S&P 500 is a stock index which is based on the prices of 500 public American companies selected by a committee. The stocks are chosen so that they represent the American economy. The S&P 500 is commonly used as a benchmark for what returns an investor may expect when exposed to the risk of the stock market.

The return of a stock (disregarding dividends) or an index is how much its price changes over a period of time $t-1$ to t. It is the fraction,

$$R_t = \frac{P_t - P_{t-1}}{P_{t-1}},$$

where P_t is the price at time t. We shall consider daily returns where a point in time t is at the close of the stock market on day t. Then P_t is the closing price of the stock/index on day t and R_t is the

return during that day. If an investor invests an amount A at time 0, the amount at time T will be $A(1+R_1)(1+R_2)\cdot\ldots\cdot(1+R_T)$.

The log-return on day t is the natural logarithm of $(1+R_t)=\frac{P_t}{P_{t-1}}$. Given a sample of returns, one can reduce the skewness of their distribution by transforming them to log-returns. Making the distribution of the returns more symmetric will also make it closer to the normal distribution. Thus, using log-returns is particularly useful if assuming normality in a model. In this thesis, we set y_t to be the log-return of the S&P 500 index over day t.

1.1.2 Markov Chain

Markov processes can be defined in continuous and discrete time but we shall focus on discrete time Markov processes referred to as Markov chains. The following theory draws heavily upon Stroock (2014).

Let a stochastic process $\{s_t\}_{t\geq 1}$ satisfy the Markov property

$$P(s_t \mid s_1, s_2, \ldots, s_{t-1}) = P(s_t \mid s_{t-1}), \text{ if } P(s_1, s_2, \ldots, s_{t-1}) > 0. \tag{1}$$

The values which can be attained by the Markov chain are called states and comprises the *state-space*. Assuming the set of states is finite, one can conveniently write it as $\mathcal{S}=\{1,2,\ldots,K\}$ where the states are labeled by integers. We can set up the transition probabilities $p_t^{ij}=P(s_t=j \mid s_{t-1}=i)$, where $i,j\in\mathcal{S}$, in a *stochastic matrix*,

$$\Gamma_t=\begin{pmatrix} p_t^{11} & p_t^{12} & \ldots & p_t^{1K} \\ p_t^{21} & p_t^{22} & \ldots & p_t^{2K} \\ \vdots & \vdots & \ddots & \vdots \\ p_t^{K1} & p_t^{K2} & \ldots & p_t^{KK} \end{pmatrix},$$

where each row sum up to one. Finally, let $\mathbf{p}$ be a vector containing the probabilities $P(s_1=i)$ where $i\in\mathcal{S}$. The process $\{s_t\}_{t\geq 1}$ satisfying the above is a Markov chain on the finite state-space $\mathcal{S}$ with initial distribution given by $\mathbf{p}$ and transition probability matrix Γ_t.

Before moving on we state the definition of a sojourn which is relevant to Markov chains as it is of interest how long the process tends to remain in each state.

Definition 1 (Rubino and Sericola, 1989). *A sojourn of X in B is $X_m, X_{m+1}, \ldots, X_{m+k}$ where $k\geq 1$, $X_m, X_{m+1}, \ldots, X_{m+k-1}\in B$ and $X_{m+k}\notin B$. If in addition, $m>0$ and $X_{m-1}\notin B$, then the sojourn begins at time m, finishes at time $m+k$ and lasts k.*

1.1.3 Hidden Markov model

Restricting ourselves to the discrete time and finite state-space case, we can define a Hidden Markov model (HMM) as a bivariate stochastic process $\{(y_t,s_t)\}_{t\geq 1}$ where $\{s_t\}_{t\geq 1}$ is an unobservable "hidden" Markov chain and y_t is a random variable such that the conditional distribution of y_t given s_t is completely determined by s_t (Cappé et al. 2006; p. 1-4). The system being modeled by a HMM can be

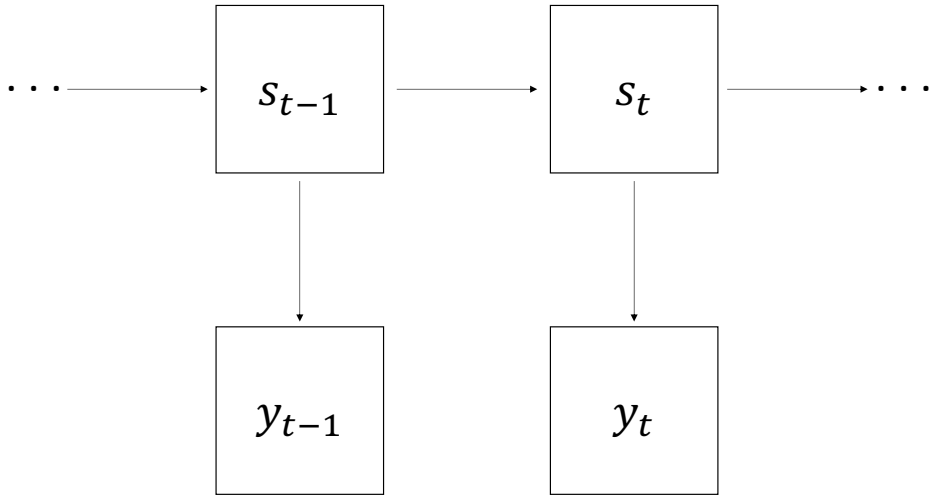


Figure 1: Schematic diagram of the system being modeled by the HMM.

thought of as an underlying sequence of states $s_1, s_2, \ldots$ where each element s_t in the state sequence generates an output y_t (see Figure 1).

Due to the states being hidden, all inference about the model has to be based on observations of y with the associated states as latent variables. When modeling a system as a HMM, one may be interested in making inference about the initial distribution $\mathbf{p}$; the transition probability matrix Γ_t for each point in time; the conditional distribution of Y; and the underlying state sequence generating the observed values in a sample.

1.1.4 Statistical learning

In *supervised learning*, the goal is to predict a response variable Y with a set of features, or predictors, $X = \{X_1, X_2, \ldots, X_p\}$. Essentially, it is about estimating an assumed function f such that $Y = f(X) + \epsilon$ where ϵ is the residual; the difference between the observed and the fitted value. It is assumed to be random and distributed around zero. This estimation is based on observations of (X, Y). If Y is discrete, the problem of estimating f is a classification problem. If continuous, it is a regression problem.

In *unsupervised learning*, on the other hand, we have observations of X but not of a response variable labeling the data. In this case we do not aim to estimate a function f as we have no observations to relate to X. The goal is rather to group or to find an informative way to visualize the data (James et al., 2013; p. 373-374). One of the main methods, or class of methods, in unsupervised learning is clustering. It involves assigning (grouping) objects to clusters based on similarity between the objects (Hastie et al. 2009; p. 501-503). If the goal is to predict the state sequence in a HMM given a time series of observations of the output variable, i.e. label each observation with a state, one could exploit the assumed dissimilarity between observations associated with different states (and similarity between

observations associated with the same state) to cluster the data into one cluster per state (when the state space is finite).

The similarity between observations in one cluster can for instance be that they are close to one another with respect to squared Euclidean distance in the feature space. That is the case for one of the more popular and well known clustering methods. The K-means clustering algorithm is initialised by randomly placing K cluster centers in the feature space. After initialization, it assigns each observation to its nearest cluster (in Euclidean distance) after which it replaces each cluster center by the centroid of the observations assigned to that cluster. This is iterated until the algorithm converges in some sense (for example when the cluster centers no longer change from one iteration to the next). The result is K clusters in which the within-cluster variance is minimised. The algorithm for jump estimation of HMMs is closely related to K-means clustering (Nystrup et al., 2020b). When the Euclidean norm is used, it could in fact be seen as an extension of K-means; adding a jump penalising regularisation term to the objective function.

2 Estimating a two state HMM with time-varying transition probabilities applied to the S&P 500

Diebold et al. (1994) proposed a HMM in which the transition probabilities are logistic functions of exogenous variables. They also developed an EM-algorithm to find the parameters maximising the likelihood of the HMM. We shall describe the model in Section 2.1 as well as outline their EM-algorithm in Section 2.2. Afterwards, we apply it to the S&P 500 and some corresponding exogenous variables which may be appropriate for the model. To choose which variables to include, we perform forward step wise variable selection.

2.1 The Model

Let $\{(y_t, s_t)\}_{t\geq 1}$ be a HMM as described in Section 1.1.3. We set the daily log-returns of the S&P 500 index to be $\{y_t\}_{t\geq 1}$ with the underlying Markov chain $\{s_t\}_{t\geq 1}$. We let the sequence of explanatory variables $\{\mathbf{x}_t\}_{t\geq 1} = \{(1, x_{t_1}, x_{t_2}, \ldots, x_{t_{k-1}})'\}_{t\geq 1}$ determine the transition probabilities of the Markov chain at each point $t = 1, 2, \ldots$ in discrete (daily) time where $k-1$ is the number of exogenous variables included in the model. With some additional assumptions, we get the model of Diebold et al. (1994).

Let the state space be comprised by two states, i.e. $K = 2$, and assume that the probability matrix of the Markov chain satisfies,

$$\Gamma_t = \begin{pmatrix} p_t^{11} & p_t^{12} \\ p_t^{21} & p_t^{22} \end{pmatrix} = \begin{pmatrix} \frac{e^{\mathbf{x}_{t-1}'\beta_1}}{1+e^{\mathbf{x}_{t-1}'\beta_1}} & 1 - p_t^{11} \\ 1 - p_t^{22} & \frac{e^{\mathbf{x}_{t-1}'\beta_2}}{1+e^{\mathbf{x}_{t-1}'\beta_2}} \end{pmatrix} \tag{2}$$

with $p_t^{ij} = P(s_t = j | s_{t-1} = i)$ where $i, j \in \{1, 2\}$ and $\beta = (\beta_1, \beta_2)' = (\beta_{1_0}, \beta_{1_1}, \ldots, \beta_{1_{k-1}}, \beta_{2_0}, \beta_{2_1}, \ldots, \beta_{2_{k-1}})' \in \mathbb{R}^{2k}$. Denote $\rho = P(s_1 = 2)$. Then, the initial distribution of this two-state Markov chain is given by ρ. Finally, we assume that the variables y_t for $t = 1, 2 \ldots$ and $i = 1, 2$ satisfy the conditional distribution,

$$y_t | s_t = i \overset{i.i.d.}{\in} N(\mu_i, \sigma_i) \iff f(y_t | s_t = i; \alpha) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(\frac{-(y_t - \mu_i)^2}{2\sigma_i^2} \right), \tag{3}$$

where $\alpha = (\alpha_1, \alpha_2) = (\mu_1, \sigma_1, \mu_2, \sigma_2)$. Then the inference is to estimate $\theta = (\alpha, \beta, \rho)$. These assumptions are not realistic and make the model a simplification of the very complex truth. However, the model may be sufficiently accurate for our purpose.

2.2 Estimating the parameters

Let $y = \{y_t\}_{t=1}^{T}$ and $x = \{\mathbf{x}_t\}_{t=1}^{T-1}$ be observations of the variables included in the model and construct the design matrix,

$$X = \begin{pmatrix} 1 & x_{1_1} & x_{1_2} & \dots & x_{1_{k-1}} \\ 1 & x_{2_1} & x_{2_2} & \dots & x_{2_{k-1}} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{T-1_1} & x_{T-1_2} & \dots & x_{T-1_{k-1}} \end{pmatrix} \tag{4}$$

of the exogenous variables. We aim to find the maximum likelihood estimate (MLE) of θ. If we would have observed the state sequence $s = \{s_t\}_{t=1}^{T}$, the goal would be to find θ which maximises the ***complete data likelihood***,

$$f(y, s|X; \theta). \tag{5}$$

Optimising the complete likelihood is is equivalent to optimising the complete log-likelihood,

$$\log f(y, s|X; \theta), \tag{6}$$

as the logarithm is a monotone transformation.

However, we cannot observe s in a HMM and thus we may not directly maximise the complete data log-likelihood. We only have information which lets us maximise the ***incomplete data log-likelihood***,

$$\log f(y|X; \theta) = \log \sum_{s_1=1}^{2} \sum_{s_2=1}^{2} \dots \sum_{s_T=1}^{2} f(y, s|X; \theta) \tag{7}$$

which is not feasible in practice due to the expression being very complex (the logarithm acts outside the sum). The EM-algorithm (Algorithm 1) with s as the ***latent*** (unobserved) variable aims to find a θ which maximises the incomplete data likelihood; thus finding the MLE. One can show that $\log f(y|X; \theta^{(j)})$ is increasing (but not necessarily strictly) for each iteration $j = 1, 2, \dots$ of the algorithm (Hastie et al., 2009; p. 276-277) but it does not always converge to the global maximum.

Algorithm 1: The EM-Algorithm

1. Initialise: Pick a θ^0 to initialise the algorithm.
2. E-Step: Construct $\mathcal{Q}(\theta, \theta^{(j-1)}) = \mathbb{E}_{s|y,X;\theta^{(j-1)}} \log f(y, s|X; \theta)$.
3. M-Step: Find $\theta^{(j)} = \arg\max_{\theta} \mathbb{E}_{s|y,X;\theta^{(j-1)}} \log f(y, s|X; \theta)$.
4. Iterate steps 2 and 3 for $j = 1, 2, 3 \dots$ until convergence.

Typically, one would iterate until the algorithm convergences in θ or the incomplete log-likelihood.

The rather tedious procedure for performing step 2 and 3 which was developed by Diebold et al. (1994) is described in great detail in their paper. We outline that method here.

Before describing how to go about constructing the expectation in the E-step, we remind the reader about the multivariate conditional density $f(u_1, u_2, \dots, u_m | u_{m+1}, u_{m+2}, \dots, u_n) = \frac{f(u_1, u_2, \dots, u_n)}{f(u_{m+1}, u_{m+2}, \dots, u_n)}$ when $f(u_{m+1}, u_2, \dots, u_n) \neq 0$ and $m \in \{1, 2, \dots, n\}$. This is widely accepted and derivation is omitted here. The variables $u_1, u_2, \dots, u_n$ can be continuous, mixed or discrete. In the latter case, f is replaced with P.

With this formula; the Markov property of $\{s_t\}_{t=1}^T$; and the independence of the variables $y_1|s_1, y_2|s_2, \dots, y_T|s_T$, we get an expression for the complete data likelihood (5),

$$\begin{aligned}
f(y, s|X; \theta) &= f(y_T, s_T | y_{1:T-1}, s_{1:T-1}, X; \theta) f(y_{1:T-1}, s_{1:T-1} | X; \theta) \\
&= \dots = f(y_1, s_1 | X; \theta) \prod_{t=2}^{T} f(y_t, s_t | y_{1:t-1}, s_{1:t-1}, X; \theta) \\
&= f(y_1 | s_1, X; \theta) P(s_1 | \rho) \prod_{t=2}^{T} f(y_t | s_{1:t}, y_{1:t-1}, X; \theta) P(s_t | y_{1:t-1}, s_{1:t-1}, X; \theta) \\
&= f(y_1 | s_1; \alpha) P(s_1 | \rho) \prod_{t=2}^{T} f(y_t | s_t; \alpha) P(s_t | s_{t-1}, \mathbf{x}_{t-1}; \beta),
\end{aligned}$$

where conditioning on X and the parameters should not be confused with conditioning on random variables. The notations $y_{1:t}$ and $s_{1:t}$ denotes the elements in the sequences y and s from time one up until time t. It follows that the complete data log-likelihood (6) is,

$$\begin{aligned}
\log f(y, s|X; \theta) &= \log f(y_1 | s_1; \alpha) + \log P(s_1 | \rho) + \sum_{t=2}^{T} \Big\{ \log f(y_t | s_t; \alpha) + \log P(s_t | s_{t-1}, x_{t-1}; \beta) \Big\} \\
&= \mathbf{1}_{\{s_1 = 1\}} \Big[\log f(y_1 | s_1 = 1; \alpha_1) + \log(1 - \rho) \Big] + \mathbf{1}_{\{s_1 = 2\}} \Big[\log f(y_1 | s_1 = 2; \alpha_2) + \log \rho \Big] \\
&+ \sum_{t=2}^{T} \Big\{ \mathbf{1}_{\{s_t = 1\}} \log f(y_t | s_t = 1; \alpha_1) + \mathbf{1}_{\{s_t = 2\}} f(y_t | s_t = 2; \alpha_2) \\
&+ \mathbf{1}_{\{s_t = 1, s_{t-1} = 1\}} \log(p_t^{11}) + \mathbf{1}_{\{s_t = 2, s_{t-1} = 1\}} \log(p_t^{12}) \\
&+ \mathbf{1}_{\{s_t = 2, s_{t-1} = 2\}} \log(p_t^{22}) + \mathbf{1}_{\{s_t = 1, s_{t-1} = 2\}} \log(p_t^{21}) \Big\},
\end{aligned}$$

where $\mathbf{1}$ is the indicator function. Taking the expectation of the complete data log-likelihood with respect

to $s|y, X; \theta^{(j-1)}$ we get,

$$\begin{aligned}
\mathbb{E}_{s|y,X;\theta^{(j-1)}} \log f(y, s|X;\theta) = & P\left(s_1 = 1 \middle| y, X, \theta^{(j-1)}\right)\left[\log f\left(y_1 \middle| s_1 = 1; \alpha_1\right)\right. \\
& \left. + \log(1-\rho)\right] + P\left(s_1 = 2 \middle| y, x, \theta^{(j-1)}\right)\left[\log f\left(y_1 \middle| s_1 = 2; \alpha_2\right) + \log\rho\right] \\
& + \sum_{t=2}^{T}\Bigg\{P\left(s_t = 1 \middle| y, X, \theta^{(j-1)}\right)\log f\left(y_t \middle| s_t = 1; \alpha_1\right) \\
& \quad + P\left(s_t = 2 \middle| y, X, \theta^{(j-1)}\right)\log f\left(y_t \middle| s_t = 2; \alpha_2\right) \\
& \quad + P\left(s_t = 1, s_{t-1} = 1 | y, X, \theta^{(j-1)}\right)\log\left(p_t^{11}\right) \\
& \quad + P\left(s_t = 2, s_{t-1} = 1 | y, X, \theta^{(j-1)}\right)\log\left(p_t^{12}\right) \\
& \quad + P\left(s_t = 1, s_{t-1} = 2 | y, X, \theta^{(j-1)}\right)\log\left(p_t^{21}\right) \\
& \quad + P\left(s_t = 2, s_{t-1} = 2 | y, X, \theta^{(j-1)}\right)\log\left(p_t^{22}\right)\Bigg\},
\end{aligned}$$

as $\mathbb{E}[\mathbf{1}_A] = P(A)$ where A is an event. The E-step of the EM-algorithm therefore amounts to evaluate the marginal and joint *smoothed* probabilities, $P\left(s_t = i|y, X; \theta^{(j-1)}\right)$ and $P\left(s_t = k, s_{t-1} = l|y, X; \theta^{(j-1)}\right)$, for $i, l, k = 1, 2$. The reader is referred to Diebold et al. (1994) for the details about how these probabilities are computed.

In the subsequent M-step, the constructed expectation is maximised with respect to θ. Diebold et al. (1994) found that one can directly find solve for $\alpha^{(j)}$ and $\rho^{(j)}$. Finding $\beta_i^{(j)}$ requires a linear Taylor approximation of $p_t^{ii}(\beta_i)$ centered at $\beta_i^{(j-1)}$. Consequently, the expectation is not maximised exactly and the error bound will be largest as β change the most from one iteration to the next. As the algorithm begins to converge, this second order error bound becomes minute. For detailed derivations of the solution to the optimisation problem, the reader is again referred to Diebold et al. (1994). The closed form solutions to the parameters of $\theta^{(j)}$ that are computed in the M-step are,

$$\begin{aligned}
\mu_i^{(j)} =& \frac{\sum_{t=1}^{T} y_t P\left(s_t = i|y, X; \theta^{(j-1)}\right)}{\sum_{t=1}^{T} P\left(s_t = i|y, X; \theta^{(j-1)}\right)}, \\
\sigma_i^{(j)} =& \sqrt{\frac{\sum_{t=1}^{T}\left(y_t - \mu_i^{(j)}\right)^2 P\left(s_t = i|y, X; \theta^{(j-1)}\right)}{\sum_{t=1}^{T} P\left(s_t = i|y, X; \theta^{(j-1)}\right)}}, \\
\rho^{(j)} =& P\left(s_1 = 2|y, X; \theta^{(j-1)}\right), \\
\beta_i^{(j)} =& \left(\sum_{t=2}^{T} \mathbf{x}_{t-1} P\left(s_{t-1} = i|y, X; \theta^{(j-1)}\right)\frac{\partial p_t^{ii}(\beta_i)}{\partial \beta_i}\right)^{-1} \times \\
& \left(\sum_{t=2}^{T} \mathbf{x}_{t-1}\left\{P\left(s_t = i, s_{t-1} = i|y, X; \theta^{(j-1)}\right) - P\left(s_{t-1} = i|y, X; \theta^{(j-1)}\right)\left[p_t^{ii}\left(\beta_i^{(j-1)}\right) - \frac{\partial p_t^{ii}(\beta_i)}{\partial \beta_i}\beta_i^{(j-1)}\right]\right\}\right),
\end{aligned}$$

where $\frac{\partial p_i^{ii}(\beta_i)}{\partial \beta_i} = \left(\frac{\partial p_i^{ii}(\beta_i)}{\partial \beta_{i_0}}, \frac{\partial p_i^{ii}(\beta_i)}{\partial \beta_{i_1}}, \ldots, \frac{\partial p_i^{ii}(\beta_i)}{\partial \beta_{i_{k-1}}} \right)$ is evaluated at $\beta_i^{(j-1)}$.

2.3 Model selection

With a given set of $k-1$ exogenous variables, there are 2^{k-1} possible models as described above to choose from by including a few, all or none of them. One could fit every possible model, evaluate the quality of each and select the best. Due to computational limitations however, we choose our model by forward selection (James et al., 2013; p. 207) instead as follows:

1. Fit the null model $\mathcal{M}_0$ without any exogenous variables.
2. For $j = 1, \ldots, k-1$: fit one model for each of the $k-j$ variables not included in model $\mathcal{M}_{j-1}$ which includes that variable in addition to those in $\mathcal{M}_{j-1}$. Denote the best of these models by $\mathcal{M}_j$.
3. Choose the best of the models $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_{k-1}$.

To perform the forward selection, we first need to introduce some way to evaluate how good the models are in relation to each other. The log-likelihood function (7) works when comparing models with the same amount of parameters (degrees of freedom). However, the likelihood tends to increase with the degrees of freedom making it badly suited for comparing models which differs in that regard; favouring more complex, possibly overfitted, models. Taking this into account, we use the measurements in the following definition which penalises model complexity.

Definition 2 *Let $\mathcal{L}(\hat{\theta})$ be the maximised likelihood of a model with parameters θ; let $k = \dim(\theta)$ be the amount of parameters estimated; and n the sample size. Then, the Akaike Information Criterion is* $\text{AIC} = 2k - 2\log \mathcal{L}(\hat{\theta})$ *and the Bayesian Information Criterion is* $\text{BIC} = \log(n)\dim(\theta) - 2\log \mathcal{L}(\hat{\theta})$.

When using AIC or BIC to compare a set of models, the model which give the smallest value is considered to be the best. The values are useful in comparing models but say nothing about them in absolute terms. AIC and BIC look similar but are derived and motivated in different ways. AIC aims to select the model which best approximates the "true model" of the data by choosing the model minimising the (estimated) expected information loss (Burnham and Andersson, 2002; p. 60-64). On the other hand, BIC aims to select the model which has the highest probability of being true given the data. If the "true model" is considered, BIC will select it almost surely as the sample size goes to infinity (Hastie et al., 2009; p. 233-235). In practice, for a large enough sample, if their favoured model differs, BIC will select the simpler model. Regarding which one to rely on, generally speaking, AIC is intended for comparing prediction accuracy while BIC may be more appropriate for explanatory modeling (Shmueli, 2010).

The EM-algorithm returns maximising a $\hat{\theta}$ (although not necessarily the global max). Moreover, we

can compute the likelihood $\mathcal{L}(\theta)$ of the model for the observed (incomplete) data as follows,

$$\begin{aligned}\mathcal{L}(\theta) &= f(y|X;\theta) = f(y_{1:T-1}|X;\theta)f(y_T|y_{1:T-1},X;\theta)\\ &= \ldots = f(y_1|X;\theta)\prod_{t=2}^{T} f(y_t|y_{1:t-1},X;\theta) \overset{1}{=} f(y_1|X;\theta)\prod_{t=2}^{T} f(y_t|y_{1:t-1},X_{1:t-1};\theta)\\ \implies \log\mathcal{L}(\theta) &= \log\left(\sum_{s_1=1}^{2} f(y_1|s_1;\theta)f(s_1|\theta)\right) + \sum_{t=2}^{T}\log f(y_t|y_{1:t-1},X_{1:t-1};\theta),\end{aligned}$$

where $X_{1:t-1}$ is the matrix comprised by the first $t-1$ rows of X. The values $f(y_1|s_1;\theta)$ and $f(s_1|\theta)$ can be calculated directly, given θ; and $f(y_t|y_{1:t-1},X_{1:t-1};\theta)$ for $t = 2,3,\ldots,T$ are each computed during the E-step of the EM-algorithm developed by Diebold et al. (1994). Hence, we may evaluate $\mathcal{L}\left(\theta^{(j-1)}\right)$ during every iteration (j) in the algorithm. We compute the maximised likelihood during the last iteration before the EM-algorithm terminates and get AIC and BIC.

2.4 Exogenous variables considered

We consider four different exogenous variables which could be seen as indicators of market sentiment. The variables are,

1. 3-Month Treasury Bill rate (DTB3),
2. CBOE Volatility Index (VIX),
3. 10-year minus 3-month Treasury yields (T10Y3M),
4. TED Spread (TED).

One could of course consider additional variables which may be well suited for our model. However, we limit ourselves to these four.

2.4.1 3-Month Treasury Bill (DTB3)

The 3-Month Treasury Bill rate is the annual yield on US government debt obligations with 3-month maturity. Due to its almost nonexistent risk, it is frequently used to measure the theoretical "risk free rate", that is, the annual return of the risk free asset. As investor uncertainty increase, demand for risk free assets such as the 3-Month T-Bill typically increase. Consequently, their price increase and their yield rates decrease.

2.4.2 CBOE Volatility Index (VIX)

The VIX is an index indicating the market's expectation of future market volatility; implied volatility. It is derived from the inputs used to compute prices of options with S&P 500 as the underlying asset. High values of VIX are often associated with high uncertainty and investor fear.

[1] This equality is intuitively clear but justified in Appendix A.

2.4.3 10-year minus 3-month Treasury yields (T10Y3M)

The 10-year minus 3-month Spread is the difference between the 10-Year and 3-month Treasury yields (10-Year minus 3-Month). Under normal circumstances, this difference is positive as investors are prepared to pay more for the most liquid asset of the two (the one with shorter maturity). If this spread is negative however, the famous (or infamous) yield curve is inverted. Historically, an inverted yield curve has in almost all cases been followed by a recession in the US economy.

2.4.4 TED spread (TED)

The TED spread is the 3-month LIBOR rate minus the 3-month T-Bill rate. The LIBOR is a benchmark rate based on the interest rates on loans between major banks. A large TED spread is considered to indicate high credit risk on these loans and vice versa. One could therefore hypothesise that a high TED spread would be associated with uncertainty on the financial markets.

2.5 Data

The daily closing prices of the S&P 500 and exogenous variables mentioned above is downloaded from FRED Economic Data (Federal Reserve Bank St. Louis, 2020). The data is taken from the time period 2012-03-26 to 2020-03-06 (With one additional day for the S&P 500 (2012-03-23) to get the daily log-return on the first day). Due to holidays, there are some days with no data. In very few of these cases, there are data missing for a subset of the variables. These days are simply removed from the data set. It is assumed that whatever effect this may have on the model is negligible. The log-returns for each day are computed before removing days with missing data in other variables so that there will not be any outliers introduced by this (if computing the return over two days instead of one).

The total sample size is 1950. Due to the availability of financial data, one could increase the sample size effortlessly and possibly make better inference. However, because this EM-algorithm is quite computationally intensive, a larger sample size would not be feasible for the author.

2.6 Preprocessing data

The noisy nature of financial time series (Abu-Mostafa & Atiya, 1996) motivates smoothing of the time series representing the exogenous variables. This aims to prevent noise and outliers affecting our model. We shall consider two ways of doing this eventually resulting in two different models. In the first approach, we perform Simple exponential smoothing (SES) on the time series. The degree of smoothing will be set by minimising the one step ahead prediction error; giving a reasonable degree of smoothing assuming that there are some structure in the series. In the second approach, we shall resort to Locally estimated scatterplot smoothing (LOESS). There, the degree of smoothing will be set manually so that the smoothed curves appear to follow the major trends in the time series. This is a very informal way of doing it. However, ensuring that we filter out much variation will be beneficial for our simulation study; we may better investigate the jump estimator's robustness to noise in exogenous variables (See section 3.1).

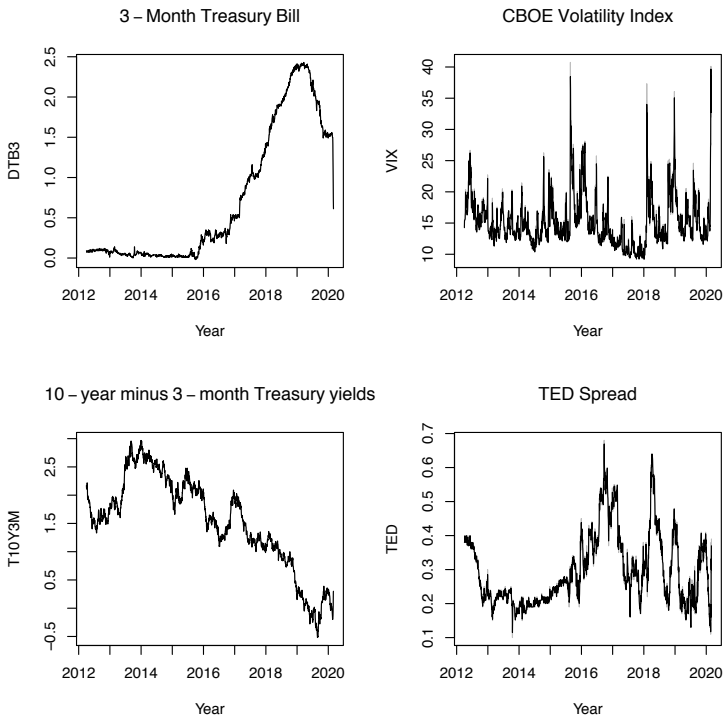


Figure 2: SES smoothed time series of exogenous variables.

2.6.1 Approach (a): SES

SES is a simple but common method for time series smoothing and prediction. Given a univariate time series $\{x_t\}_{t=1}^{T}$ (In this case, not to be confused with row t in the design matrix), we transform it recursively according to

$$z_t = \alpha x_t + (1-\alpha) z_{t-1}$$

starting from $z_1 = x_1$. If we would like to predict x_{t+1}, SES predicts $\hat{x}_{t+1} = z_t$. The parameter α controls the degree of smoothing. It is between 0 and 1 and determines the proportion of which the fitted value is based on the most recent value. We choose α by minimising the one step ahead prediction error over the entire time series in our sample (separately for each exogenous variable). The prediction error is in this case defined as the mean of the losses from each prediction. It is desirable to use a robust loss function due to the noise and possible outliers in the time series. We consider the Huber (1964) loss,

$$L_\delta(\epsilon) = \begin{cases} \frac{1}{2}\epsilon^2 & \text{if} \quad |\epsilon| \le \delta, \\ \delta(|\epsilon| - \frac{1}{2}\delta), & \text{otherwise}, \end{cases} \tag{8}$$

where $\epsilon = x - \hat{x}$ is the residual of a prediction which we assume is unbiased (so $\mathbb{E}(\epsilon) = 0$). This loss combines the square and absolute loss. For "normal" values of ϵ ($\le \delta$) it is the square loss. However,

for errors with abnormally large magnitude it is the absolute loss which is less sensitive to outliers. A rule of thumb is to classify an outlier as one which is more than three standard deviations away from the mean. Therefore, we set $\delta = 3\texttt{std}(\epsilon)$ where we estimate the standard deviation from the set of residuals obtained when doing one step ahead prediction over the entire time series. As $\epsilon_t = x_t - \hat{x}_t = x_t - z_{t-1}$, the prediction error is $\frac{1}{T-1}\sum_{t=2}^{T} L_\delta(\epsilon_t)$ where $\delta = 3\texttt{std}(\epsilon_2, \ldots, \epsilon_T)$.

We consider an equidistant grid of α's spaced by 0.01. The prediction error is computed as described above for each of these parameters and the α which minimises the error is chosen. This procedure is done separately for each exogenous variable resulting in Table 1 and the time series in Figure 2. The smoothing resulting from this approach is minimal but we do see a slight reduction of spikes in the time series.

Table 1: Best α for simple exponential smoothing.

	DTB3	VIX	T10Y3M	TED
α	0.98	0.84	0.96	0.78

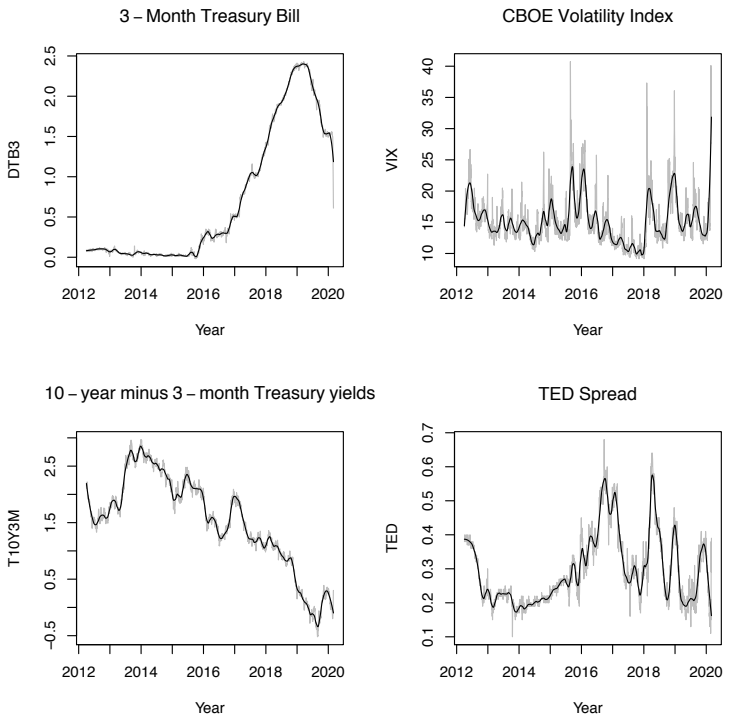


Figure 3: LOESS smoothed time series of exogenous variables.

2.6.2 Approach (b): LOESS

LOESS involves fitting a polynomial to a specified amount of nearest neighbors to each point by weighted least squares. To describe LOESS, we again consider a univariate time series $\{x_t\}_{t=1}^{T}$. Then LOESS maps

x_t to $\hat{x}_t = p(t)$ setting

$$p = \arg\min_{p \in \mathcal{P}^n} \sum_{i \in \mathcal{N}_t} W(i)(x_i - p(i))^2$$

where $\mathcal{P}^n$ is the set of polynomials of degree n; $\mathcal{N}_t$ is the set of indices of the k nearest neighbours to t; and $W(i)$ is a weight function which ensures that the fit is more influenced by closer points than those further away. Commonly it is the tri-cube weight function $W(i) = \left(1 - |d_i|^3\right)^3$ where $|d_i|$ is the distance from i to t normalized to $[0, 1]$. We use these weights and quadratic polynomials ($n = 2$). The degree of smoothing, *span*, is determined by the amount of points considered in each neighbourhood divided by the length of the entire time series. Increasing this fraction will give smoother results. This method does not assume one function explaining the entire time series. Furthermore, it is beneficial compared to some other kernel methods in terms of bias at the boundaries (Hastie et al. 2009; p.192-198).

We choose the span manually in such a way that the fitted curve appears to capture major, longer term signals of the time series while filtering out sudden, sharp moves. The chosen span is 0.05 for all exogenous variables; resulting in all variables being equally forward and backward looking. It appears to have the largest effect on the time series of VIX and TED spread which are more volatile. The smoothed and raw versions of the time series are illustrated in Figure 3.

2.6.3 Z-score standardisation

Before being inputted into the EM-algorithm, the variables are standardised according to Z-score standardisation

$$x_{t_j}^{stand} = \frac{x_{t_j} - \texttt{mean}(x_j)}{\texttt{std}(x_j)} \tag{9}$$

which simplifies the initial choice of initial θ^0 (in particular β^0) as all exogenous variables will have sample mean and standard deviation zero and one respectively. Here, x_j is the time series of the j'th variable. The smoothed and standardised time series are finally used to construct the inputted design matrix X.[2] To make β more interpretable, one can unstandardise the coefficients to readjust the model to the non-standardised data. This is done as follows,

$$\begin{aligned}
\beta_{i_0} + \sum_{j=1}^{k-1} \beta_{i_j} x_{j_t} = \beta_{i_0}^{stand} + \sum_{j=1}^{k-1} \beta_{i_j}^{stand} x_{t_j}^{stand} &= \beta_{i_0}^{stand} + \sum_{j=1}^{k-1} \beta_{i_j}^{stand} \frac{x_{t_j} - \texttt{mean}(x_j)}{\texttt{std}(x_j)} \\
&= \left(\beta_{i_0}^{stand} - \sum_{j=1}^{k-1} \beta_{i_j}^{stand} \frac{\texttt{mean}(x_j)}{\texttt{std}(x_j)} \right) + \sum_{j=1}^{k-1} \frac{\beta_{i_j}^{stand}}{\texttt{std}(x_j)} x_{j_t} \\
&\implies \begin{cases} \beta_{i_0} = \beta_{i_0}^{stand} - \sum_{j=1}^{k-1} \beta_{i_j}^{stand} \frac{\texttt{mean}(x_j)}{\texttt{std}(x_j)}, \\ \beta_{i_j} = \frac{\beta_{i_j}^{stand}}{\texttt{std}(x_j)}, \quad j = 1, 2, \ldots, k-1. \end{cases}
\end{aligned}$$

[2] Including the appropriate variables for each step in the forward selection and each approach (a) and (b).

Table 2: BIC and AIC of the models (with included variables) for each step of the forward selection in approach (a). Bold entries identify the best values.

Model	AIC	BIC
$\mathcal{M}_0$	−13631.1366	−13592.1075
$\mathcal{M}_1^a$ (VIX)	−13770.5723	**−13720.3921**
$\mathcal{M}_2^a$ (VIX+TED)	**−13775.5127**	−13714.1813
$\mathcal{M}_3^a$ (VIX+TED+DTB3)	−13772.2440	−13699.7614
$\mathcal{M}_4^a$ (All)	−13771.5600	−13687.9262

2.7 Execution on S&P 500

We do the forward selection as described in Section 2.3 separately for approaches (a) and (b). The EM-algorithm is implemented in R with $\texttt{max}(|\theta^{(j)} - \theta^{(j-1)}|) < 10^{-6}$ as stopping criterion; the iterations are stopped as the largest change in the parameters from one iteration to the next is smaller than 10^{-6}. Each time we run the algorithm, we initialise it with a θ^0 in which each parameter is chosen randomly and uniformly on reasonable intervals.

2.7.1 Execution: approach (a)

We fit the models to y and the SES transformed time series in X according to the forward selection scheme. The selected models with AIC and BIC at each step are reported in Table 2. The different criteria disagree on which model is best; with AIC preferring a more complex model than BIC.

Our purpose is partly to investigate which exogenous variables explain y through the model best. However, in the forthcoming simulation study we shall generate simulated state sequences sequentially according to the transition probabilities p_t^{ij} predicted by the selected $\mathbf{x}_{t-1}$. This suggests relying on AIC rather than BIC. Therefore, we select $\mathcal{M}_2^a$ but note that BIC favours $\mathcal{M}_1^a$ which includes only VIX. The estimated parameters of $\mathcal{M}_2^a$ are,

$$\mu_1 = 0.0006, \;\; \sigma_1 = 0.0045,$$

$$\beta_1^{stand} = (0.4593, -3.6164, 0.1056)',$$

$$\mu_2 = -0.0001, \;\; \sigma_2 = 0.0127,$$

$$\beta_2^{stand} = (-0.6627, 2.3837, -0.4951)',$$

$$\rho = 1.$$

We invert the standardisation and get the coefficients $\beta_1 = (14.2172, -0.9257, 1.0119)'$ and $\beta_2 =$

Table 3: BIC and AIC of the models (with included variables) for each step of the forward selection in approach (b). Bold entries identify the best values.

Model	AIC	BIC
$\mathcal{M}_0$	−13631.1366	−13592.1075
$\mathcal{M}_1^b$ (VIX)	**−13714.1890**	**−13664.0087**
$\mathcal{M}_2^b$ (VIX+T10Y3M)	−13714.0036	−13652.6721
$\mathcal{M}_3^b$ (VIX+T10Y3M+TED)	−13711.6521	−13639.1695
$\mathcal{M}_4^b$ (All)	−13708.8858	−13625.2521

$(-8.5240, 0.6102, -4.7441)'$. [3]

2.7.2 Execution: approach (b)

The models are fit to y and the LOESS transformed time series in X. The best model with AIC and BIC values at each step are reported in Table 3. $\mathcal{M}_1^b$ is selected by both AIC and BIC and the parameters of $\mathcal{M}_1^b$ are,

$$\mu_1 = 0.0010, \quad \sigma_1 = 0.0046,$$

$$\beta_1^{stand} = (2.0709, -1.3377)',$$

$$\mu_2 = -0.0007, \quad \sigma_2 = 0.0123,$$

$$\beta_2^{stand} = (1.2640, 1.7318)',$$

$$\rho = 0.$$

After inverting the standardisation we get $\beta_1 = (8.5539, -0.4265)'$ and $\beta_2 = (-7.1289, 0.5521)'$. Figure 4 shows how the transition probabilities depend on the selected variable VIX.

2.8 Discussion

Doing the selection in another way, for example by backward selection, would possibly result in other models. Different degrees of time series smoothing of the exogenous variables and even initialisation of the EM-algorithm also have the potential to affect the results; $\mathcal{M}_2^b$ is a very close runner up when considering AIC. However, a key observation in Tables 2 and 3 is that both AIC and BIC increase quite substantially when letting the transition probabilities depend on variance in VIX ($\mathcal{M}_1^a$ and $\mathcal{M}_1^b$) as opposed to keeping them fixed ($\mathcal{M}_0$).

[3] The coefficients are given in the same order as the variables in the table with the first coefficient being the constant term.

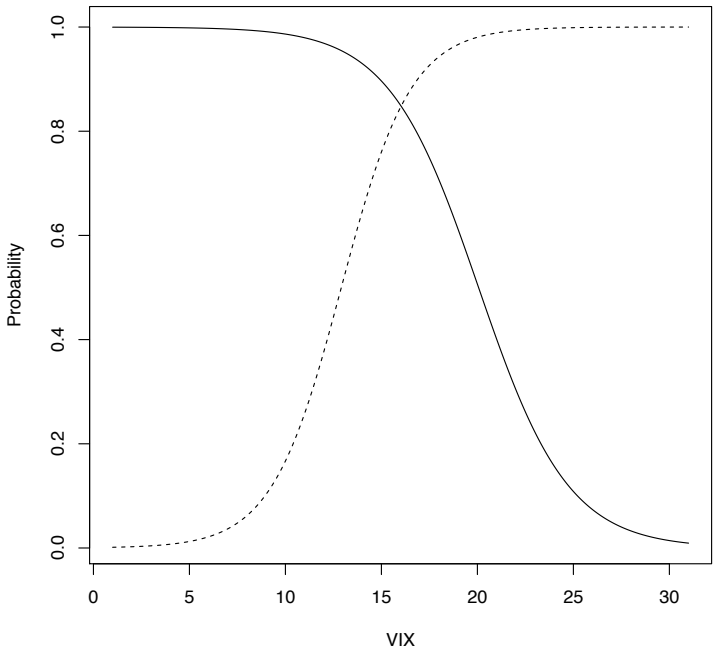


Figure 4: Graph of p^{11} (solid) and p^{22} (dashed) from $\mathcal{M}_1^b$ as functions of VIX.

The states of the selected models, $\mathcal{M}_2^a$ and $\mathcal{M}_1^b$, represent a less volatile bull market (state one) and a more volatile bear market (state two). Moreover, as intuition suggests, the probability of transitioning to state two (bear market) increases with VIX in both models. Vice versa holds for state one. Conversely, according to $\mathcal{M}_2^a$, an increasing TED spread would increase the probability of transitioning to, or staying in the bull market (and vice versa). This is unexpected considering that which is mentioned in section 2.4.4. However, the standardised coefficient associated with the TED spread is much smaller than that of VIX, setting the TED spread's influence on the model to be relatively small.

Comparing approaches (a) and (b), we see that the obtained AIC and BIC values are better in (a) than in (b) indicating that the models in (a) have more predictive and explanatory power than those in (b). Seeing the SES and LOESS smoothed variables as different variables and performing the selection considering all of these eight variables; we would at a guess still end up selecting model $\mathcal{M}_1^a$ or $\mathcal{M}_2^a$. The lower likelihood of the models in (b) compared to (a) can be explained by the LOESS smoothing being too severe; removing much relevant information from the exogenous variables. On the other hand, the SES smoothing, with smoothing parameter chosen automatically, does little to filter potential noise and outliers.

It would be interesting to see if one could improve the models further by altering the methods in the preprocessing stage. We add that winsorization (remove and replace extreme values) was tried to get rid of outliers more than three standard deviations away from the mean. This did not seem to have any significant effect on the resulting model. One could, instead, probably improve the model by considering

different degrees of smoothing to filter out the ”right” amount of noise.

Comparing SES with LOESS, we note that SES is backward looking while LOESS is, in addition, forward looking. Consequently, the LOESS smoothed time series does not reflect the available data at each point in time. Although it is not what we observe in our results (due to the degree of smoothing by LOESS being severe), this might inflate the computed likelihoods.[4] SES is therefore a less questionable method even though one has access to the entire interval when fitting the models. This does not make the resulting model, $\mathcal{M}_1^b$, from approach (b) appear to be very good as, in addition to its lower likelihood, it is based on LOESS smoothed explanatory variables. However, it is useful for our simulation study as it allows for some interesting experiments based on inputting either raw or smoothed data to the jump estimator when applied to series generated according to $\mathcal{M}_1^b$. For that end, the advantage of LOESS is that the smoothed time series is not lagged which is the case for SES. In our simulation study, we can see the excess variation in the raw data as noise, being distributed around the LOESS curve considered as the true signal (See Figure 3).

To conclude this section, we have two models, $\mathcal{M}_2^a$ and $\mathcal{M}_1^b$, to facilitate the simulation study in Section 4. The models from approach (a) appears to explain the data better than that from approach (b). However, in this setting, both approaches results in VIX being deemed, of those considered, the most important explanatory variable for this class of HMMs. Furthermore, a HMM with time-varying transition probabilities improves upon the null model.

[4] Discussed with and remarked by P. Nystrup.

3 Jump estimation of HMMs

Algorithm 2: Jump estimation of HMM (Nystrup et al., 2020b).

Input: Time series $u = \{u_1, u_2, \dots, u_T\}$; the number of latent states K; and initial state sequence $s^0 = \{s_1^0, s_2^0, \dots, s_T^0\}$.

1. Construct a set of standardised features z from the time series u.
2. Iterate for $i = 1, 2, \dots$ until $s^{(i)} = s^{(i-1)}$.
   (a) $\theta^{(i)} = \arg\min_{\theta} \sum_{t=1}^{T} \ell\left(z_t, \theta_{s_t^{(i-1)}}\right)$ (model fitting).
   (b) $s^{(i)} = \arg\min_{s} \left\{ \sum_{t=1}^{T-1} \left[\ell\left(z_t, \theta_{s_t}^{(i)}\right) + \lambda \mathbf{1}_{\{s_t \neq s_{t+1}\}} \right] + \ell\left(z_T, \theta_{s_T}^{(i)}\right) \right\}$ (state sequence fitting).
3. Compute the average sojourn duration and distributional parameters for each state.

Output: HMM parameters and prediction of latent states.

Nystrup et al. (2020b) proposed a novel method for learning a HMM by minimising the objective function of Bemporad et al. (2018) for fitting jump models;

$$\sum_{t=1}^{T-1} \left[\ell(z_t, \theta_{s_t}) + \lambda \mathbf{1}_{\{s_t \neq s_{t+1}\}} \right] + \ell(z_T, \theta_{s_T}), \qquad \lambda \geq 0. \tag{10}$$

where ℓ is a loss function, in this case $\ell(z_t, \theta_{s_t}) = \|z_t - \theta_{s_t}\|_2^2$. It requires finding the minimising state sequence $s = \{s_1, s_2, \dots, s_T\}$ and parameters $\theta = \{\theta_1, \theta_2, \dots, \theta_K\}$ given an observed time series of features $z = \{z_1, z_2, \dots, z_T\}$. Seeing the elements of θ as cluster centers, the similarities to K-means with objective function analogous to $\sum_{t=1}^{T} \|z_t - \theta_{s_t}\|_2^2$ are clear (Hastie et al. 2009; p. 509-510).

Algorithm 2, which aims to minimise the objective function (10), is just as described by Nystrup et al. (2020b) apart from step 3 where computation of transition probabilities has been replaced with computation of average sojourn duration for each state. This is so because the transition probabilities are not assumed to be constant in our case. The optimisation problem in Item 2a is solved by finding $\theta^{(i)}$ such that $\frac{\partial}{\partial \theta_j} \sum_{t=1}^{T} \ell(z_t, \theta_{s_t^{(i-1)}}) \big|_{\theta_j = \theta_j^{(i)}} = 0$ for $j = 1, 2, \dots, K$. By the strict convexity of the squared Euclidean norm, this is indeed the minimising θ. Using the differentiation rules of matrix calculus we have,

$$\begin{aligned} \frac{\partial}{\partial \theta_j} \sum_{t=1}^{T} \|z_t - \theta_{s_t^{(i-1)}}\|_2^2 &= \frac{\partial}{\partial \theta_j} \sum_{t=1}^{T} \left(z_t - \theta_{s_t^{(i-1)}}\right)' \left(z_t - \theta_{s_t^{(i-1)}}\right) = \frac{\partial}{\partial \theta_j} \sum_{t=1}^{T} \left(z_t' z_t - 2 z_t' \theta_{s_t^{(i-1)}} + \theta'_{s_t^{(i-1)}} \theta_{s_t^{(i-1)}}\right) \\ &= \sum_{t:\, s_t^{(i-1)} = j} \left(2\theta'_{s_t^{(i-1)}} - 2 z_t'\right). \end{aligned}$$

Setting this expression to zero we get the solution $\theta_j^{(i)} = \frac{1}{N_j} \sum_{t:\, s_t^{(i-1)} = j} z_t$, where N_j is the number of elements

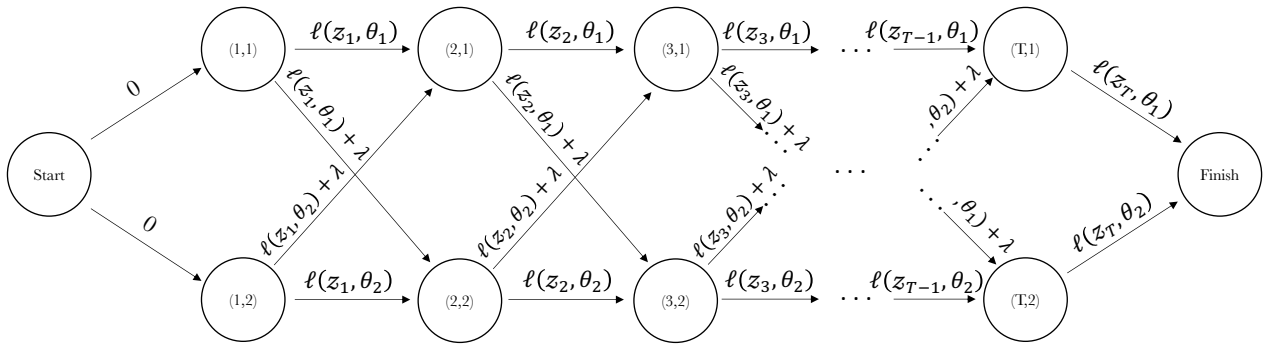


Figure 5: Graph illustrating the optimisation problem solved by dynamic programming when $K = 2$. The minimising state sequence is the least costly path from start to finish.

in $s^{(i-1)}$ being equal to j. This is the centroid of the observations in z being assigned to state j during the previous iteration. Nystrup et al. (2020b) find $s^{(i)}$ in item 2b by the dynamic programming method of Bellman (1957). Firstly, define

$$
\begin{aligned}
&V(T,s) = \ell(z_T,\theta_s),\\
&V(t,i) = \ell(z_t,\theta_i) + \min_j\Big[V(t+1,j) + \lambda\mathbf{1}_{\{i\neq j\}}\Big],\ \ t = T-1,\dots,1.
\end{aligned}
$$

Given θ, the sequence s minimising the objective function is

$$
\begin{aligned}
&s_1 = \arg\min_j V(1,j),\\
&s_t = \arg\min_j\Big[V(t,j) + \lambda\mathbf{1}_{\{s_{t-1}\neq j\}}\Big],\ \ t = 2,\dots,T.
\end{aligned}
$$

This finds the least costly path through T nodes with the choice between K nodes at each point in time where $V(t,i)$ is the best path from node (t,i) to finish and the cost of going from (t,i) to $(t+1,j)$ is $\ell(z_t,\theta_i) + \lambda\mathbf{1}_{\{i\neq j\}}$. Finding this path which sums up to the least cost is equivalent to minimizing the objective function with respect to s. This is illustrated in Figure 5 for the case when the state space is comprised by two states.

Nystrup et al. (2020b) run Algorithm 2 from ten different initial state sequences generated by K-means++; choosing the output which gives the lowest objective value. K-means++ initialisation spreads out the initial cluster centers randomly in a certain way (Arthur and Vassilvitskii, 2006). This is motivated by the fact that the algorithm risk getting stuck in local minimums of the objective function. They also terminate the algorithm after a maximum of ten iterations or as the objective value changes by less than 10^{-6} from one iteration to the next. Here, we shall use this stopping criterion and, in the manner as mentioned above, initialise Algorithm 2 with ten different state sequences generated by K-means++ initialisation. Finally, the features z are standardised by Z-score standardisation (9). Although the algorithm does both estimation and prediction, it is still referred to as the jump estimator.

Table 4: For each point in time $t = l, \ldots, T$, we construct the features from time series x and y.

Features considered for the jump estimator.
1. Observation: y_t
2. Absolute change: $\lvert y_t - y_{t-1} \rvert$
3. Previous absolute change: $\lvert y_{t-1} - y_{t-2} \rvert$
4. Centered mean: $\texttt{mean}(y_{t-l+1}, \ldots, y_t)$
5. Centered standard deviation: $\texttt{std}(y_{t-l+1}, \ldots, y_t)$
6. Left mean: $\texttt{mean}(y_{t-l+1}, \ldots, y_{t-\frac{l}{2}})$
7. Left standard deviation: $\texttt{std}(y_{t-l+1}, \ldots, y_{t-\frac{l}{2}})$
8. Right mean: $\texttt{mean}(y_{t-\frac{l}{2}+1}, \ldots, y_t)$
9. Right standard deviation: $\texttt{std}(y_{t-\frac{l}{2}+1}, \ldots, y_t)$
10. Exogenous variable j: x_{t_j}
11. Square root of x_{t_j}: $\sqrt{x_{t_j}}$
12. Natural logarithm of x_{t_j}: $\log \mathrm{x}_{t_j}$
13. Reciprocal of exponential x_{t_j}: $\exp(-\mathrm{x}_{t_j})$
14. Probability p^{11}: $e^{x_t\beta_1}/(1 + e^{x_t\beta_1})$
15. Probability p^{22}: $e^{x_t\beta_2}/(1 + e^{x_t\beta_2})$

3.1 Input and features

We allow u in Algorithm 2 to be a multivariate time series and consider,

$$u = \{(x_1, y_1), (x_2, y_2), \ldots, (x_T, y_T)\},$$

where x represents the exogenous variables and y is, as before, the log-returns of the S&P 500. We use the same backward looking features derived from y as Nystrup et al. (2020a) with window lengths $l = 6$ and $l = 14$. In addition to those, we consider x and a few transformations of x (see Table 2: 10-13). Adding transformations of x to the set of features can have several consequences. Firstly, a transformed feature may reveal patterns not recognisable from the original data. Secondly, adding transformations increases the impact of x to the euclidean distances in the objective function (10).[5] This should lead to a greater influence of the information from x on the predicted state sequence. It may be desirable to do this because we want to compare performance when including versus excluding features based on x. If their influence is minimal, it might prevent us from getting any significant results even if x contribute with information which would be useful. However, we do not claim that the feature spaces which we shall use are optimal with regards to x. Weighing the information from x in relation to y is interesting but beyond the scope of this thesis.

As explained by Zheng et al. (2019) who originally motivated a version of the "vanilla" features derived from y, features 4-9 are intended to contain information about the conditional distributions characterising each state. Furthermore, the absolute changes might be useful in detecting regime switches

[5] Adding the same feature twice, would be (disregarding initialisation) equivalent to weighing that feature's contribution to the euclidean distances by a factor of two.

Table 5: Different jump estimators for approach (a): $\mathcal{M}_2^a$ and their feature spaces.

Name	Features	Dimension of feature space	Inputted data
Vanilla^a	1-9 ($l = 6$ & $l = 14$)	15	S&P 500 log-returns
Extended^a	1-13 ($l = 6$ & $l = 14$)	23	S&P 500 log-returns; raw VIX and raw TED spread

Table 6: Different jump estimators for approach (b): $\mathcal{M}_1^b$ and their feature spaces.

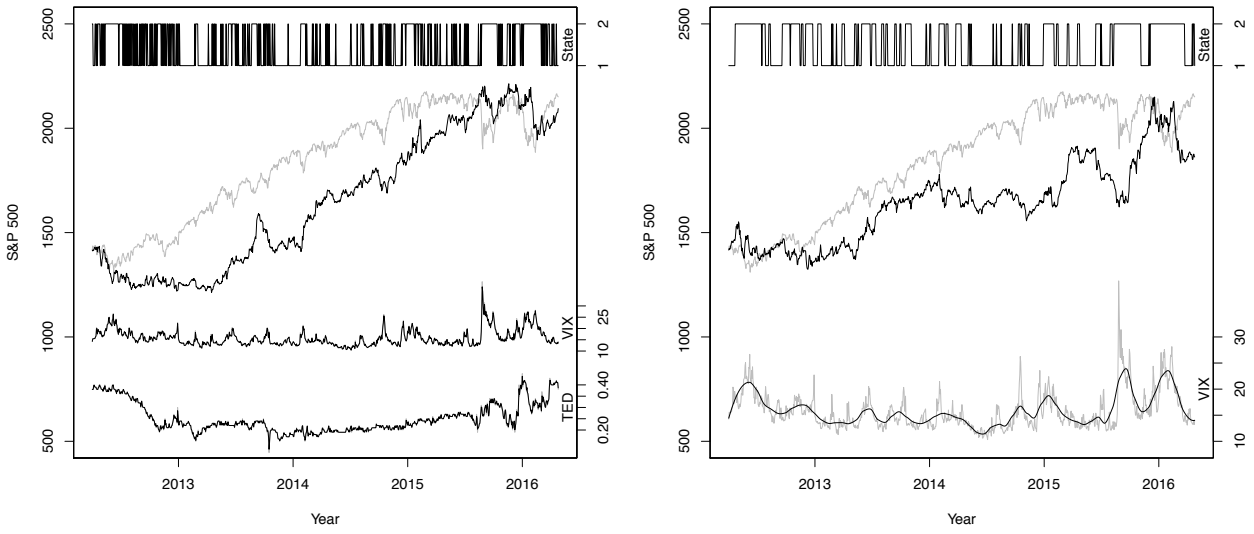
Name	Features	Dimension of feature space	Inputted data
Vanilla^b	1-9 ($l = 6$ & $l = 14$)	15	S&P 500 log-returns
Extended^b	1-13 ($l = 6$ & $l = 14$)	19	S&P 500 log-returns and raw VIX
Oracle^b	1-9 ($l = 6$ & $l = 14$) & 14-15 (added twice)	17	S&P 500 log-returns and raw VIX
Ext. smooth^b	1-13 ($l = 6$ & $l = 14$)	19	S&P 500 log-returns and LOESS VIX
Ora. smooth^b	1-9 ($l = 6$ & $l = 14$) & 14-15 (added twice)	17	S&P 500 log-returns and LOESS VIX

before the change of conditional distribution is reflected in the local means and standard deviations. The features related to the exogenous variables (10-15) should contain information about the probabilities of staying in or transitioning to each state at each point in time in the assumed HMM. We hypothesise that this information is beneficial for jump estimation. In contrast to most of the features based on y, the exogenous features are not backward looking.

We shall use several different input data and feature spaces resulting in different jump estimators. They are named and described in Tables 5 and 6 and are intended to be applied to time series simulated according to the HMMs from approaches (a) and (b) respectively. The HMM from approach (a), $\mathcal{M}_2^a$, includes twice as many exogenous variables as $\mathcal{M}_1^b$ from approach (b). Therefore, the feature space for Extended^a is of higher dimensions than for Extended^b.

The explanatory variable (VIX) in $\mathcal{M}_1^b$ is heavily smoothed by LOESS. We can use this to investigate the jump estimators robustness to noise in the exogenous variables. To that end, we consider different estimators inputted with either raw or smoothed exogenous time series. Moreover, we define the Oracle^b estimator which replaces features 10-13 by 14 and 15; each added twice to keep the proportion of features based on x the same as in Extended^b.[6] Features 14 and 15 are the probabilities of staying in the respective states (and $1 - P\{\text{transition}\}$) when going from one day to the next (the exact probabilities when LOESS smoothed VIX is used). Of the estimators in Table 6, Oracle smooth^b is the jump estimator with features which, intuitively, would be most relevant for $\mathcal{M}_1^b$. This is however cheating as it is not realistic to know that information. The feature spaces and jump estimators using the exogenous variable(s) will be referred to as the extended feature spaces and estimators.

[6] This is done in order to make the estimators more comparable; taking in consideration what was mentioned in the first paragraph of this section.



(a) $\mathcal{M}_2^a$ (b) $\mathcal{M}_1^b$

Figure 6: Simulated and real S&P 500 indexes (black and grey respectively) according to models $\mathcal{M}_2^a$ (a) and $\mathcal{M}_1^b$ (b) with state sequences on top and exogenous time series at the bottom.

4 Simulation Study

We compare the performance of the different jump estimators in a simulation study. The daily log-returns of S&P 500 are simulated using the models, $\mathcal{M}_2^a$ and $\mathcal{M}_1^b$, selected in Section 2.7. The simulated data allow us to compare the predicted state sequence to the true state sequence. Each simulated series is created by first generating a state sequence according to the transition probabilities at each point in time in Γ_t; given by the parameters of the selected HMM and exogenous variables from a randomly chosen period of time between 2012-03-26 and 2020-03-06 of desired length. The first element in the sequence is generated according to the initial state distribution ρ. Given the state sequence, we sample each simulated daily log-return y_t from the conditional Gaussian distribution with parameters determined by s_t. Figure 6 illustrates the simulated S&P 500 generated according to $\mathcal{M}_2^a$ and $\mathcal{M}_1^b$. We can get the index values from the log-returns by accumulatively multiplying the index with exp(log-return) starting from an appropriate value.

The state classification performance is measured in *balanced accuracy* (BAC) which is the average of the true positive rates for each state $\frac{TP}{TP+FN}$, where TP is the number of true positives which was correctly classified and FN is the number of true positives which was incorrectly classified (For state one, positive is 1 and negative is 2. Vice versa for state two.). Due to the risk of label switching when clustering, i.e. the algorithm labels the cluster really associated with state one as two and vice versa; the states are labeled in such a way that these accuracy rates are maximised which is simply done when only considering two states. As Nystrup et al. (2020b) pointed out, BAC is a suitable measure of accuracy as it does not get inflated when a classifier take advantage of an imbalanced data set. If 95 percent of the observations

belong to state 1, then a classifier which classifies all observations to the same state (in this case that would be a jump estimator with large enough jump penalty λ) would have 95 percent accuracy according to accuracy defined by $\frac{\#\text{correct classifications}}{\#\text{total classifications}}$.

We remark that we are more concerned about the performance of the different estimators in relation to each other than their performance in absolute terms. Our goal in this study is to investigate if including exogenous variables improves the jump estimator. Therefore, the relative performance is what we are interested in.

4.1 Choosing the jump penalty λ

Before comparing the different jump estimators we need to decide which jump penalties to use. All estimators should be optimised with respect to the jump penalty λ in every setting. As BAC is used to measure the performance of the jump estimators, we optimise them by maximising BAC with respect to λ. BAC is a random quantity (function of the data $\mathcal{D}$ and initial state sequence, s^0, generated by K-means++ which are both random) so we aim to find the λ which maximises $\mathcal{B}(\lambda) = \mathbb{E}(\text{BAC}|\lambda)$. We can estimate this by $\hat{\mathcal{B}}(\lambda) = \frac{1}{n}\sum_{i=1}^{n} \text{BAC}_\lambda^i$ where $\text{BAC}_\lambda^1, \text{BAC}_\lambda^2, \ldots, \text{BAC}_\lambda^n$ is a sample of observations of BAC for a given λ which we get by applying the jump estimators to simulated time series.

We discretise the problem by considering a grid of jump penalties on the natural logarithmic scale and choose the maximising λ out of these. Thirteen gridpoints are used for most estimators apart from some which require more to the left.[7] Setting $n = 1000$ and computing $\hat{\mathcal{B}}$ for each jump estimator and penalties separately on time series of length 250, 500 and 1000 in each setting ($\mathcal{M}_2^a$ and $\mathcal{M}_1^b$), we get Figures 7 and 8 and Tables 7 and 8 of approximated optimal jump penalties. To get a sense of the accuracy of $\hat{\mathcal{B}}_{1000}(\lambda)$ we compute the sample standard error (SE) $\frac{s}{\sqrt{1000}}$, where s is the sample standard deviation of BAC. This is an estimate of how much, on average, we may expect $\hat{\mathcal{B}}_{1000}$ to deviate from the true mean and is shown in the figures around each curve.

In most cases, the estimators does not seem to be terribly sensitive to which jump penalty is used. The exception is Vanilla^b which, for the longest time series, has a clearly defined maximum. In general, we see a slight improvement in BAC when increasing λ from zero; favouring jump estimation over K-means clustering.

[7] A finer grid was initially considered but the improvement in BAC when using more grid points is negligible.

Table 7: Approximated optimal λ for each jump estimator applied to time series simulated according to $\mathcal{M}_2^a$.

	250	*500*	*1000*
Vanilla^a	0	0	20
Extended^a	0	0	1

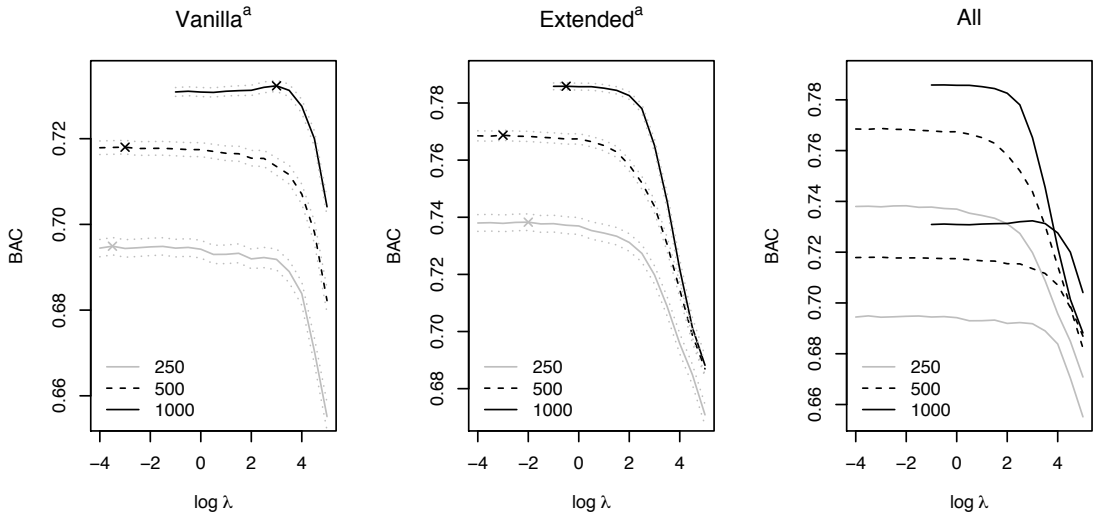


Figure 7: BAC (± standard error) as a function of λ for the jump estimators estimators (a) and time series of different lengths generated by $\mathcal{M}_2^a$.

Table 8: Approximated optimal λ for each jump estimator applied to time series simulated according to $\mathcal{M}_1^b$.

	250	*500*	*1000*
Vanilla^b	4	12	33
Extended^b	12	7	7
Oracle^b	0	4	7
Ext. smooth^b	1	3	4
Ora. smooth^b	1	2	1

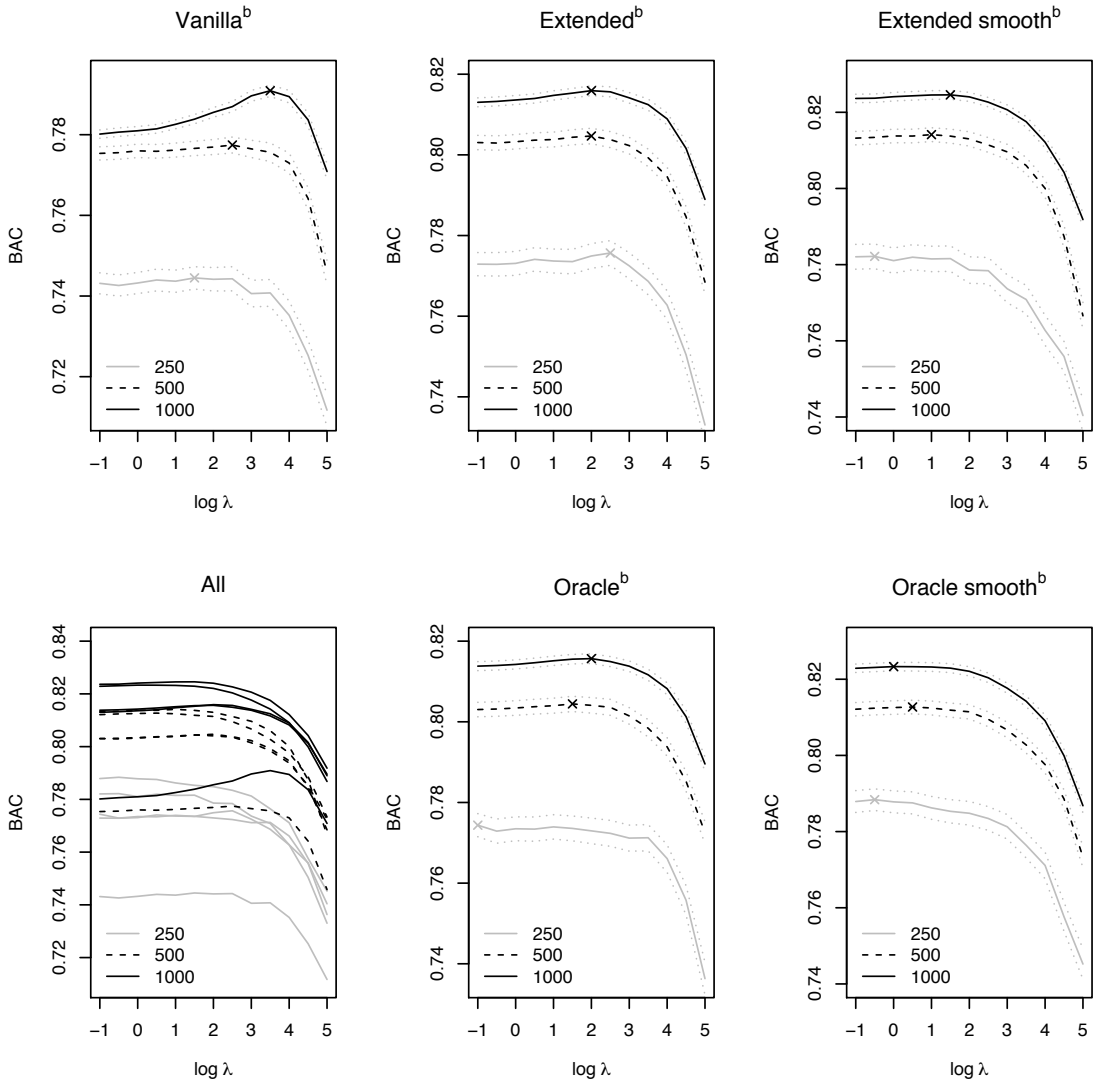


Figure 8: BAC (± standard error) as a function of λ for the jump estimators (b) and time series of different lengths generated by $\mathcal{M}_1^b$.

Table 9: Mean (and standard deviation) of the estimated parameters, average sojourn durations and accuracies based on 1000 simulated series of different lengths according to $\mathcal{M}_2^a$. Bold entries identify the best estimates between the jump estimators.

	μ_1	σ_1	μ_2	σ_2	Sojourn 1	Sojourn 2	Accuracy 1	Accuracy 2	BAC
250									
True^a	0.0006	0.0045	−0.0001	0.0127	6.5143 (6.5076)	2.6662 (0.8115)	0.9347 (0.0376)	0.7511 (0.1041)	0.8430 (0.0466)
Vanilla^a	0.0007 (0.0009)	**0.0059** (0.0009)	−0.0005 (0.0028)	**0.0118** (0.0019)	**19.9020** (17.9337)	**7.6799** (2.7092)	**0.8284** (0.0780)	0.5614 (0.1266)	0.6949 (0.0640)
Extended^a	**0.0006** (0.0007)	**0.0059** (0.0011)	**−0.0001** (0.0019)	0.0111 (0.0021)	29.5597 (29.4313)	15.6159 (9.7462)	0.8063 (0.1055)	**0.6702** (0.1702)	**0.7383** (0.0916)
500									
True^a	0.0006	0.0045	−0.0001	0.0127	6.1625 (2.4163)	2.6924 (0.5013)	0.9463 (0.0216)	0.7693 (0.0622)	0.8578 (0.0316)
Vanilla^a	0.0007 (0.0007)	0.0057 (0.0006)	−0.0005 (0.0021)	**0.0120** (0.0010)	**19.8843** (10.1203)	**7.3932** (1.8051)	**0.8505** (0.0531)	0.5855 (0.0943)	0.7180 (0.0503)
Extended^a	**0.0006** (0.0005)	**0.0055** (0.0007)	**−0.0001** (0.0015)	0.0114 (0.0013)	25.8899 (13.8286)	13.4258 (7.0501)	0.8225 (0.0968)	**0.7149** (0.1113)	**0.7687** (0.0561)
1000									
True^a	0.0006	0.0045	−0.0001	0.0127	5.7720 (0.9286)	2.8160 (0.2347)	0.9481 (0.0131)	0.7836 (0.0320)	0.8658 (0.0185)
Vanilla^a	**0.0007** (0.0004)	0.0059 (0.0005)	−0.0004 (0.0014)	**0.0120** (0.0007)	58.3594 (14.9180)	24.1286 (6.6309)	**0.8631** (0.0449)	0.6016 (0.0976)	0.7323 (0.0418)
Extended^a	**0.0007** (0.0003)	**0.0054** (0.0005)	**−0.0003** (0.0011)	**0.0120** (0.0007)	**25.0400** (5.4762)	**12.1738** (2.5925)	0.8598 (0.0370)	**0.7119** (0.0739)	**0.7859** (0.0374)

4.2 Comparison results

Figures 7 and 8 show how the estimators including exogenous variables dominate the vanilla estimators in terms of BAC. We report the estimated parameters, average sojourn durations and accuracies for each jump estimator in Tables 9 and 10 based on 1000 time series of different lengths. The values are those obtained using the optimising jump penalties in Tables 7 and 8 for each case. Tables 9 and 10 also show the true parameters and mean of true average sojourns of the data generating models. The true accuracies are based on the state sequences predicted by the Viterbi algorithm (Viterbi, 1967) with the true model parameters and transition probabilities at each point in time to find the most likely state sequence given the observed data. It computes $s^* = \arg\max_s f(s|y, X; \mathcal{M}) = \arg\max_s f(s, y|X; \mathcal{M})$, where X is the observed exogenous data and $\mathcal{M}$ is the HMM (in this case $\mathcal{M}_2^a$ or $\mathcal{M}_1^b$). The Viterbi algorithm is analogous to that which is used to compute the minimising state sequence in Algorithm 2 but with reversed time order of operations (Nystrup et al., 2020b) and other costs (or rewards in this case) between the nodes.

4.2.1 Setting (a)

As can be seen in Table 9, the true average sojourn durations of state one is more than twice the length of those of state two in the time series simulated by $\mathcal{M}_2^a$. It is the second state which both Vanilla^a and Extended^a find the hardest to correctly predict. All the improvement in BAC when extending the feature space stems from predicting this state and is between 4.3 and 5.4 percent (increasing with time series lengths).

Overall, Extended^a appears to be better at estimating the parameters (except σ_2) of the conditional distributions. However, both estimators overestimate σ_1 and underestimate σ_2. Concerning the average sojourns, they are greatly overestimated by both estimators and more so for Extended^a than Vanilla^a apart from the case when the vanilla estimator uses a much larger jump penalty.

Table 10: Mean (and standard deviation) of the estimated parameters, average sojourn durations and accuracies based on 1000 simulated series of different lengths according to $\mathcal{M}_1^b$. Bold entries identify the best estimates between the jump estimators.

	μ_1	σ_1	μ_2	σ_2	Sojourn 1	Sojourn 2	Accuracy 1	Accuracy 2	BAC
250									
True^b	0.0010	0.0046	−0.0007	0.0123	15.3919 (11.5295)	7.2222 (3.9810)	0.9409 (0.0531)	0.7412 (0.1987)	0.8410 (0.0891)
Vanilla^b	0.0011 (0.0007)	0.0058 (0.0010)	−0.0011 (0.0024)	**0.0116** (0.0022)	35.6395 (25.2529)	15.0610 (8.5362)	**0.8743** (0.0932)	0.6147 (0.1579)	0.7445 (0.0876)
Extended^b	**0.0010** (0.0007)	0.0058 (0.0010)	−0.0006 (0.0020)	0.0112 (0.0021)	54.0608 (36.2556)	29.7383 (19.6765)	0.8645 (0.1071)	0.6868 (0.1733)	0.7757 (0.0970)
Oracle^b	**0.0010** (0.0007)	0.0057 (0.0009)	−0.0008 (0.0021)	0.0112 (0.0020)	**29.7115** (27.7335)	**13.8654** (9.5266)	0.8702 (0.0880)	0.6786 (0.1586)	0.7744 (0.0906)
Ext. smooth^b	**0.0010** (0.0007)	**0.0056** (0.0009)	**−0.0007** (0.0019)	0.0109 (0.0023)	39.9825 (35.1917)	22.2157 (13.8798)	0.8426 (0.1215)	**0.7217** (0.1577)	0.7822 (0.1022)
Ora. smooth^b	**0.0010** (0.0007)	**0.0056** (0.0009)	**−0.0007** (0.0019)	0.0110 (0.0022)	44.8774 (40.8064)	23.1447 (14.0159)	0.8557 (0.1109)	0.7211 (0.1435)	**0.7884** (0.0897)
500									
True^b	0.0010	0.0046	−0.0007	0.0123	14.6287 (5.4068)	7.2117 (2.4916)	0.9513 (0.0305)	0.7889 (0.0958)	0.8701 (0.0461)
Vanilla^b	0.0011 (0.0004)	0.0057 (0.0006)	−0.0012 (0.0016)	**0.0120** (0.0011)	57.1085 (36.0757)	21.9371 (10.2290)	0.9070 (0.0484)	0.6479 (0.1249)	0.7774 (0.0625)
Extended^b	**0.0010** (0.0004)	0.0056 (0.0006)	**−0.0010** (0.0014)	0.0118 (0.0010)	56.9719 (32.9057)	24.1777 (11.7821)	0.9072 (0.0484)	0.7021 (0.1136)	0.8047 (0.0606)
Oracle^b	**0.0010** (0.0004)	0.0056 (0.0006)	**−0.0010** (0.0014)	0.0118 (0.0010)	51.2308 (30.3543)	21.2465 (10.3653)	0.9087 (0.0486)	0.7002 (0.1083)	0.8044 (0.0597)
Ext. smooth^b	**0.0010** (0.0004)	**0.0055** (0.0005)	**−0.0010** (0.0014)	0.0118 (0.0011)	50.8331 (33.7968)	21.9237 (10.1061)	0.9043 (0.0521)	**0.7240** (0.1063)	**0.8141** (0.0573)
Ora. smooth^b	**0.0010** (0.0004)	0.0056 (0.0005)	**−0.0010** (0.0014)	0.0118 (0.0011)	**50.5769** (35.9228)	**20.5192** (9.5897)	**0.9108** (0.0534)	0.7146 (0.1038)	0.8127 (0.0566)
1000									
True^b	0.0010	0.0046	−0.0007	0.0123	13.6458 (2.4174)	7.4531 (1.3544)	0.9504 (0.0176)	0.8148 (0.0462)	0.8826 (0.0242)
Vanilla^b	**0.0010** (0.0003)	0.0058 (0.0004)	−0.0011 (0.0010)	0.0120 (0.0007)	78.6825 (20.0044)	34.3050 (10.6923)	0.9097 (0.0389)	0.6721 (0.1012)	0.7909 (0.0449)
Extended^b	**0.0010** (0.0003)	0.0056 (0.0004)	**−0.0010** (0.0009)	0.0120 (0.0006)	50.5651 (13.1131)	23.6073 (6.7319)	0.9085 (0.0305)	0.7233 (0.0801)	0.8159 (0.0381)
Oracle^b	**0.0010** (0.0003)	0.0057 (0.0004)	**−0.0010** (0.0009)	0.0120 (0.0006)	52.0750 (13.0933)	23.9657 (6.7515)	0.9115 (0.0306)	0.7199 (0.0794)	0.8157 (0.0382)
Ext. smooth^b	**0.0010** (0.0003)	**0.0055** (0.0003)	**−0.0010** (0.0009)	0.0120 (0.0006)	48.7714 (12.8331)	23.3458 (6.8080)	0.9092 (0.0293)	**0.7399** (0.0763)	**0.8246** (0.0366)
Ora. smooth^b	0.0011 (0.0003)	0.0056 (0.0003)	−0.0011 (0.0009)	**0.0121** (0.0006)	**37.0433** (10.0273)	**16.9038** (4.7838)	**0.9167** (0.0272)	0.7299 (0.0711)	0.8233 (0.0352)

4.2.2 Setting (b)

In Table 10 we see that similarly to (a), the true average sojourns of state one lasts more or less twice as long as those in state two in the sequence generated by $\mathcal{M}_1^b$. Moreover, the extended estimators are slightly better at estimating most of the distribution parameters. As in (a), Vanilla^b prevails at estimating σ_2 but this difference diminishes with increasing time series length. Regarding the average sojourns, they are overestimated by all estimators. Using different jump penalties leads to different average predicted sojourns.

The improvement in BAC when extending the feature space comes, again, from correctly predicting state two; with Extended^b and Oracle^b improving upon Vanilla^b and the "smooth" versions of those performing even better. Adding the raw VIX to the feature space increases BAC by around 2.5- to 3%. By using the LOESS smoothed VIX instead, we increase BAC further by around 1%.

It is a bit surprising that the Oracle estimators does not beat the extended estimators in terms of BAC. Particularly when considering the "smooth" versions; where the Oracle smooth^b has the transition probabilities of $\mathcal{M}_1^b$ as features. This suggests that simply adding a few common transformations (11-13 in Table 4) of the exogenous variable to the set of features is sufficient.

The inferior performance of Extended^b and Oracle^b compared to their "smooth" counterparts is the effect which the noise in VIX has on the estimators. This noise is the quite significant amount of variation in VIX which was filtered by LOESS (see Figure 3).

4.3 Discussion

The main takeaway from Tables 9 and 10 is the improvement in classification accuracy as the feature space is extended with features from exogenous variables. In both settings (a) and (b), this improvement

is based on correctly predicting the state with the shortest average sojourn duration. A possible reason to why this is the case would be that the vanilla estimator relies more heavily on backward looking features than the extended estimators. Thus, the feature spaces of the latter estimators are to a larger extent based on current observations of the inputted variables at each point in time. This could be advantageous for correctly predicting shorter lasting sojourns.

Seeing average sojourns as parameters to be estimated, the estimates are heavily biased by all estimators. Due to them using different jump penalties in different settings and for different time series lengths, it is difficult to determine which jump estimator is best or worst in this regard based on our results. In Figure 8 in particular, we see that it is, in some cases, possible to use a smaller jump penalty without much loss in accuracy. This could be a good idea to get less biased estimates of the average sojourns. However, in practical applications it may be desirable for the jump estimator to predict longer sojourns while still maintaining accuracy. If it would react to the very short lived and sporadic regime switches demonstrated in Figure 6, that could be problematic. For example, when basing a trading strategy on the predicted states, it would lead to great transaction costs (Nystrup et al., 2020b).

The relationship between the exogenous variable(s) and the transition probabilities in $\mathcal{M}_2^a$ and $\mathcal{M}_1^b$ is such that the Markov process is more likely to transition to one state for high values of the variable(s) and the other for low values. This specific relationship is probably what allowed the jump estimator to make use of the exogenous data as it is a clustering method based on dissimilarity in (squared) Euclidean distance between feature values associated with different states. Would the coefficients related to different states in $\mathcal{M}_2^a$ and $\mathcal{M}_1^b$ instead be of the same sign, we would not expect the same improvement as is demonstrated in this simulation study.

It is encouraging to see that using the "noisy" as opposed to the smooth VIX in setting (b) only degrades the BAC by around 1% across time series lengths. This can be put in relation to the increase of around 2.5- to 3% when extending the feature space with features based on raw VIX data. It is a reassuring property of the jump estimator that it can utilise the information of the exogenous variable in spite of excess variation, which judging by Figures 3 and 6b is severe in setting (b). Whatever financial variables inputted in a real setting, would likely contain noise distorting the relevant signal. Another result demonstrating the extended jump estimator's robustness to using suboptimal features is the equal performance of Extended^b and Oracle^b. This suggests in particular that the jump estimator does not need to assume the exact shape of the curves describing the relationship between the exogenous variables and the transition probabilities. However, it would be interesting to adjust the weight of the exogenous variable in the feature space. This is not explored in this thesis as the exogenous variable's contribution to the feature spaces are set to be equal for all extended estimators in setting (b) and in setting (a) there is only one estimator using the exogenous variable.

The state sequences computed by the Viterbi algorithm are the most accurate in all cases which is not surprising as it uses the true model parameters and transition probabilities at each point in time. This suggests that it would be interesting to compare the jump estimator in this setting with the more traditional approach for learning HMMs as well; (1) use the EM-algorithm in Section 2 to estimate the parameters and predict all transition probabilities with the exogenous data; and (2) compute the most

likely state sequence using the Viterbi algorithm based on those parameters and transition probabilities. This method applied to the simulated data in settings (a) and (b) could possibly outperform the extended jump estimators. However, it would be an unfair competition as the models generating the time series in (a) and (b) are exactly those which would be assumed in this procedure. On the other hand, an interesting experiment is to change the conditional distributions to something else (such as the Student's t-distribution) when sampling the observable time series y. That would likely demonstrate one of the strengths of the jump estimator; it is robust to misspecified conditional distributions (Nystrup et al., 2020b). Furthermore, the jump estimator make less assumptions about the manner of the relationship between the exogenous variables and the transition probabilities. This is however at the cost of not making strong inference about those probabilities. Although we have used both the appropriate EM and Viterbi algorithms in this project, we could not execute these experiments in the simulation study. The reason being that the EM-algorithm is too computationally intensive for it to be feasible for the author to run it on 1000 simulated time series for each time series length in each setting.

Table 11: Sharpe ratios of different portfolios.

	Buy and hold	Vanilla	Extended
Sharpe	0.05	0.16	0.19

5 Application to real data

Before concluding, we apply the jump estimator to real data. In this case, we cannot use BAC to assess performance as the ”true” state sequence is unknown. Instead, we evaluate the performance of a ”trading strategy” based on the predicted states. The strategy is simple; hold a long position in the S&P 500 on days when the predicted state is that with positive mean return and hold no position during the other days. We compare the performance of the portfolios, Vanilla and Extended, based on the states predicted by the vanilla estimator and the extended estimator inputted with raw VIX and (in addition to vanilla features) using features 10-13 in Table 4. The vanilla and extended estimators are applied to the S&P 500 (log-returns) using the entire period in our data set. The daily returns of the two portfolios are either zero or the same as those of the index depending on which state is predicted each day. This is not at all a realistic setting as the jump estimators have access to the entire time period when predicting the states and the decision to enter or exit the market depends on future data. Furthermore, we completely disregard transaction costs and other frictions. This experiment may therefore seem ridiculous but the idea is that if the results from our simulation study are useful in practice, we should see that the Extended portfolio outperforms the Vanilla portfolio as it should be better at identifying periods with bearish and bullish market conditions.

We use the *Sharpe ratio* to assess performance but without taking the risk free interest rate into account. Then, in our case, we define the daily Sharpe ratio for a portfolio p to be

$$Sharpe := \frac{\mu_p}{\sigma_p}$$

where μ_p and σ_p is the mean and standard deviation of the daily returns of portfolio p. This ratio is optimised (on a grid as in Section 4.1) with respect to the jump penalty for each of the estimators and we get $\lambda_V^* = 0.14$ and $\lambda_{Ext}^* = 0.03$. The optimal jump penalties would, reasonably, be greater if we accounted for transaction costs as well and if the ”trading decisions” would be solely based on the available data at each point in time.

We obtain the Sharpe ratios in Table 11 along with the time series of the portfolio values in Figure 9. The Buy and hold portfolio simply follows index. We see that Extended clearly performs best. However, we emphasise that these results should not be confused with backtesting a trading strategy.

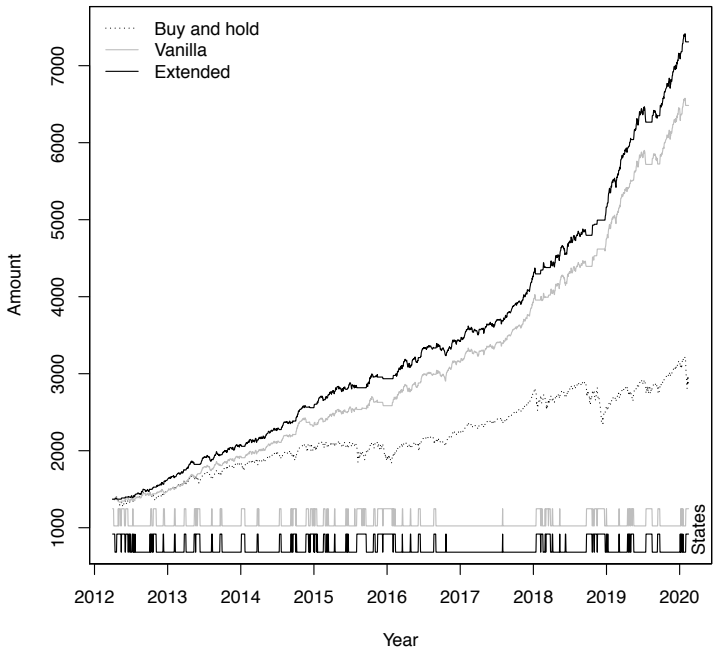


Figure 9: Accumulated amount of each portfolio starting from the appropriate price of the S&P 500 index.

6 Conclusions

In Section 2 we used the EM-algorithm to estimate two HMMs with time-varying transition probabilities depending on exogenous variables. These models were fitted to the daily log-returns of the S&P 500 with exogenous variables selected through forward selection. The two approaches for fitting the HMMs differed in the way the exogenous variables were smoothed. For the first model, we used SES with parameter chosen automatically. For the other model, we used LOESS and filtered out much variation in the time series; allowing for more interesting experiments in the subsequent simulation study. Out of the variables considered, we found that VIX was the most important explanatory variable for the assumed HMM and improved upon the null model with fixed transition probabilities.

In Section 4, we conducted a simulation study based on time series generated according to the obtained HMMs. We aimed to investigate if adding features based on exogenous data to the jump estimator would lead to an improvement. The results showed that the jump estimator's ability to correctly predict latent states was indeed significantly improved in every setting. However, the improvement was entirely based on predicting the state with, on average, shorter lasting sojourns. This could be due to the extended feature space being overall less backward looking than the vanilla space. We also saw the extended estimators demonstrating some robustness to noise in the exogenous time series. Moreover, adding a few common transformations of the included exogenous variable to the feature space appeared to be sufficient in order for the jump estimator to fully utilise the information provided by the

variable; using the ”optimal” transformations (appearing in the data generating model) instead led to no improvement.

In conclusion, we have shown that, assuming a two-state HMM with transition probabilities depending on exogenous variables, adding those variables as features to the jump estimator can improve prediction of latent states; particularly those with, on average, shorter lasting sojourns. Furthermore, the jump estimator does not appear to require any assumptions about the exact manner of the relationships between the exogenous variables and the transition probabilities apart from, possibly, each variable influencing the probabilities of transitioning to each state in opposite directions. For applications to the S&P 500, a HMM depending on VIX (and possibly TED spread) in such a way seems to be well suited. The results in Section 5, when applying the vanilla and extended estimators to real data, indicate that what we learned from our simulation study may indeed be useful in practice.

6.1 Further work

The difference in the resulting HMMs from Section 2 showed what an effect the smoothing of the exogenous time series had. Although the fitted HMMs provided us with interesting results, the data preprocessing could be done in a more informative and careful manner as it may improve the estimated model. It would also be interesting to perform the experiments which was discussed in the last paragraph of the discussion in Section 4; comparing the jump estimator with the EM-algorithm's estimates combined with the Viterbi algorithm's most likely state sequence. This needs a more efficient implementation of the EM-algorithm and more computational power than that which the author had access to. We would also like to consider additional exogenous variables for the forward selection in Section 2.

As for the jump estimator, it remains to be shown that extending the feature space with exogenous data may lead to a performance increase in an online setting as well. The features of the jump estimators used here are all backward (or current) looking. Hence, they could be applied to streaming data. We would be interested in repeating this simulation study, but instead testing the out of sample performance, particularly with the Greedy online state classifier of Nystrup et al. (2020a). A qualified hypothesis would be that the online state classifier using the extended feature space instead of the vanilla would detect state changes earlier; showing similar results as those obtained by Nystrup et al. (2020a) when adding realised volatility as features to the online classifier.

Bibliography

Y. S. Abu-Mostafa and A. F. Atiya. Introduction to financial forecasting. *Applied intelligence*, 6(3):205–213, 1996.

D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

R. Bellman. *Dynamic programming*. Princeton University Press, 1957.

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

K. P. Burnham and D. R. Anderson. *Model selection and multimodel inference: a practical information-theoretic approach*. Springer Science & Business Media, 2002.

O. Cappé, E. Moulines, and T. Rydén. *Inference in hidden Markov models*. Springer Science & Business Media, 2006.

R. Cont. Empirical properties of asset returns: stylized facts and statistical issues. 2001.

F. X. Diebold, J.-H. Lee, and G. Weinbach. "Regime Switching with Time-Varying Transition Probabilities," in *Nonstationary Time Series*. pages 283–302, 1994.

T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

P. J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics.*, 35(1):73–101, 1964.

G. James, D. Witten, T. Hastie, and R. Tibshirani. *An introduction to statistical learning*. Springer, 2013.

P. Nystrup, P. N. Kolm, and E. Lindström. Greedy online classification of persistent market states using realized intraday volatility features. 2020a.

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

G. Rubino and B. Sericola. Sojourn times in finite markov processes. *Journal of Applied Probability*, 26(4): 744–756, 1989.

G. Shmueli et al. To explain or to predict? *Statistical science*, 25(3):289–310, 2010.

D. W. Stroock. *An introduction to Markov processes*. Springer Science & Business Media, 2014.

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

K. Zheng, Y. Li, and W. Xu. Regime switching model estimation: spectral clustering hidden markov model. *Annals of Operations Research*, pages 1–23, 2019.

Appendix

A Justification of equality related to incomplete likelihood

For $t \geq 2$ we have,

$$f(y_t|y_{1:t-1}, X; \theta) = \sum_{s_1, s_2, \ldots, s_t} f(y_t, s_{1:t}|y_{1:t-1}, X; \theta) = \sum_{s_1, s_2, \ldots, s_t} f(y_t|s_{1:t}, y_{1:t-1}, X; \theta) f(s_{1:t}|y_{1:t-1}, X; \theta).$$

Clearly, the conditional distribution of y_t given s_t is independent of X (and all other conditioning variables except s_t and θ). Thus $f(y_t|s_{1:t}, y_{1:t-1}, X; \theta) = f(y_t|s_{1:t}, y_{1:t-1}, X_{1:t-1}; \theta)$. Furthermore,

$$f(s_{1:t}|y_{1:t-1}, X; \theta) = f(s_1|y_{1:t-1}, X; \theta) \prod_{j=2}^{t} f(s_j|s_{1:j-1}, y_{1:t-1}, X; \theta).$$

In our model, s_1 is only dependent on θ and s_j is only dependent on s_{j-1}, X_{j-1} and θ. Thus we have,

$$f(s_{1:t}|y_{1:t-1}, X; \theta) = f(s_1|y_{1:t-1}, X_{1:t-1}; \theta) \prod_{j=2}^{t} f(s_j|s_{1:j-1}, y_{1:t-1}, X_{1:t-1}; \theta) = f(s_{1:t}|y_{1:t-1}, X_{1:t-1}; \theta).$$

Using the above we arrive at,

$$\begin{aligned} f(y_t|y_{1:t-1}, X; \theta) &= \sum_{s_1, s_2, \ldots, s_t} f(y_t|s_{1:t}, y_{1:t-1}, X; \theta) f(s_{1:t}|y_{1:t-1}, X; \theta) \\ &= \sum_{s_1, s_2, \ldots, s_t} f(y_t|s_{1:t}, y_{1:t-1}, X_{1:t-1}; \theta) f(s_{1:t}|y_{1:t-1}, X_{1:t-1}; \theta) = f(y_t|y_{1:t-1}, X_{1:t-1}; \theta). \end{aligned}$$

Q.E.D.

Bachelor's Theses in Mathematical Sciences 2020:K9
ISSN 1654-6229

LUNFMS-4044-2020

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lth.se/