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

Petter Kolm
NYU Courant

petter.kolm@nyu.edu
https://www.linkedin.com/in/petterkolm

UC Berkeley CDAR
February 16, 2021

Our papers related to this talk

▶ Nystrup, Kolm, and Lindström (2020), "Greedy Online Classification of Persistent Market States Using Realized Intraday Volatility Features," *Journal of Financial Data Science*, 2 (3)

▶ Nystrup, Kolm, and Lindström (2021), "Feature Selection in Jump Models," *submitted paper*

Background

What are the regimes of S&P 500?
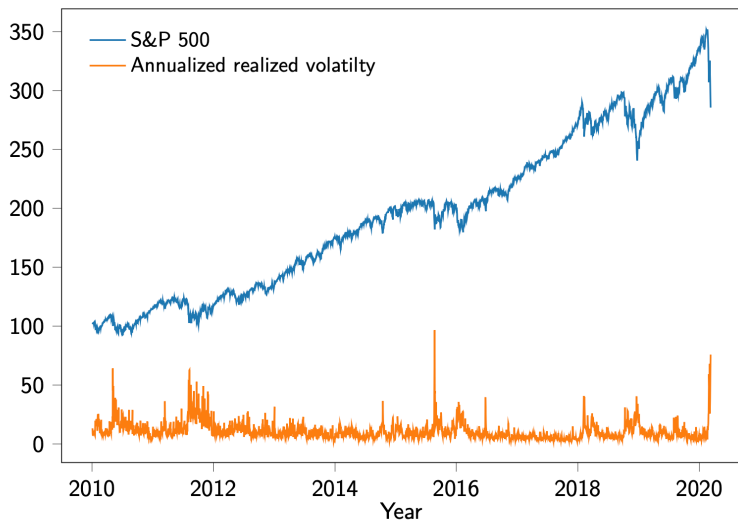
Regime switching in finance I

- ▶ Regime-switching models are used extensively in financial modeling in equities, fixed income, foreign exchange, commodities etc. where time series often exhibit
  - ▶ Heavy tails
  - ▶ Volatility clustering
  - ▶ Nonlinearities
- ▶ They are prevalent in many areas including
  - ▶ Asset allocation
  - ▶ Portfolio and risk management
  - ▶ Macroeconomic forecasting
  - ▶ Security and factor forecasting
  - ▶ High-frequency trading

Regime switching in finance II

- The popularity of the hidden Markov model (HMM) is partly due to that resulting hidden states (regimes) have meaningful interpretations
  - Risk-on and risk-off
  - Business cycles
  - Inflation and deflation
  - Periods of different waiting times in between trades

Challenges with classical HMM

- ▶ The classical HMM comes with its challenges
  - ▶ Sensitive to model specifications
    - ▶ Markov assumption
    - ▶ Misspecified conditional distributions
  - ▶ Not robust to outliers
  - ▶ Lack of temporal persistence of hidden states
  - ▶ Needs a lot of data to produce efficient estimates
  - ▶ Computationally expensive due to slow convergence
  - ▶ Nontrivial to incorporate exogenous features

What we do

1. Introduce a greedy online classifier that contemporaneously determines which hidden state a new observation belongs to
   - without the need to parse historical observations
   - without compromising temporal persistence

2. Our new classifier obtains higher accuracy and is more robust to misspecification than the correctly specified maximum likelihood estimator

3. Classification accuracy can be improved by including features that are based on intraday volatility data
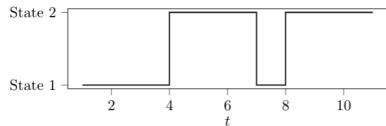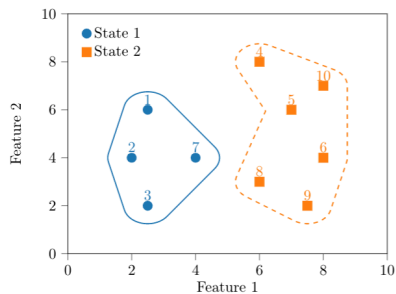
Jump models

Jump models I

"Sometimes all you need is a big leap of faith" (Sean Bean)
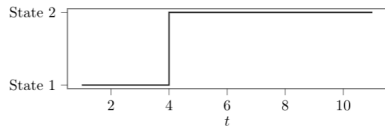
Jump models II
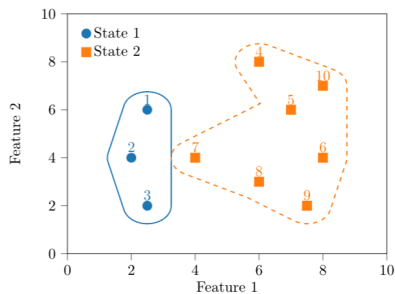

"Leap of faith yes, but only after reflection." (Soren Kierkegaard)

Jump models III

(a) K-means   (b) Jump

Illustration of the states resulting from clustering a time series of $t = 1, \ldots, 10$ observations of two features using K-means and a jump model.

Fitting jump models

Suppose $y = \{y_1, ..., y_T\}$ is a time series of T observations

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

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

over the model parameters $\theta = \{\theta_1, \ldots, \theta_K\}$ and the state sequence $s = \{s_1, \ldots, s_T\}$

Notation:
- Loss function: $\ell(y_t, \theta_{s_t}) := \|y_t - \theta_{s_t}\|^2$ (squared Euclidean distance)
- Jump penalty: λ

Features for HMM estimation

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

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

Output: Feature set $z = \{z_1, \ldots, z_T\}$ (standardized)

Jump estimation of HMM using coordinate descent

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

1. Construct a set of standardized features z from the time series y

2. Iterate for $i = 1, \ldots$

   a. Fit model parameters $\theta^i = \text{argmin}_\theta \sum_{t=1}^{T} \ell\left(z_t, \theta_{s_t^{i-1}}\right)$

   b. Fit state sequence
   $s^i = \text{argmin}_s \left\{ \sum_{t=1}^{T-1} \left[\ell\left(z_t, \theta_{s_t}^i\right) + \lambda \mathbb{I}_{s_t \neq s_{t+1}} \right] + \ell\left(z_T, \theta_{s_T}^i\right) \right\}$

3. Until $s^i = s^{i-1}$

4. Compute the transition probabilities and distributional parameters for each state

Output: HMM parameters and prediction of latent states

Fitting the state-sequence by dynamic programming

▶ Define

$$V(T, s) = \ell(z_T, \theta_s)$$
$$V(t, i) = \ell(z_t, \theta_i) + \min_j [V(t+1, j) + \lambda \mathbb{I}_{i \neq j}], \quad t = T-1, \ldots, 1$$

▶ Then, the most likely sequence of states is given by

$$s_1 = \operatorname{argmin}_j V(1, j)$$
$$s_t = \operatorname{argmin}_j [V(t, j) + \lambda \mathbb{I}_{s_{t-1} \neq j}], \quad t = 2, \ldots, T$$

▶ This becomes the Viterbi algorithm if the time order of operations is reversed

Jump estimation vs. EM

- ▶ **Complexity:** Finding the most likely sequence of states requires $O(TK^2)$ operations just like EM
- ▶ **Iterations:** Jump estimation <5 vs. EM 50–100
- ▶ Least-squares criterion does *not* rely on distributional assumptions
- ▶ **Robust** to initialization and an increasing number of states
- ▶ Estimation of parameters and state sequence *without* Markov assumption

Greedy online state classification

In practical applications, it is of critical significance to estimate the model recursively in an online fashion. The jump model can be used in this fashion

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

1. Update
   $$\mathcal{A}_t(s_t) = \min_{s_{t-1}} \left\{ \ell(z_{t-1}, \theta_{s_{t-1}}) + \mathcal{A}_{t-1}(s_{t-1}) + \lambda \mathbb{I}_{s_{t-1} \neq s_t} \right\}$$

2. Compute $\hat{s}_t = \text{argmin}_s \left\{ \ell(z_t, \theta_s) + \mathcal{A}_t(s) \right\}$

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

Simulation studies

Summary of simulation studies

We compare the new classifier with HMMs (via MLE) and spectral clustering on simulated data where the true state sequence is known

Main findings include:

► Classifier has higher accuracy in most situations, both in- and out-of-sample

► Classifier is robust to misspecification (misspecified conditional and sojourn-time distributions)

► Increasing sampling frequency of data used to estimate (volatility) features improves accuracy

Simulation study

▶ We simulate data from a two-state Gaussian HMM

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

▶ s_t is a first-order Markov chain

▶ Parameters

$$\mu_1 = .0006, \qquad \mu_2 = -.0008,$$
$$\sigma_1 = .0078, \qquad \sigma_2 = .0174,$$
$$\Gamma = \left(\begin{array}{cc} .9979 & .0021 \\ .0120 & .9880 \end{array} \right)$$
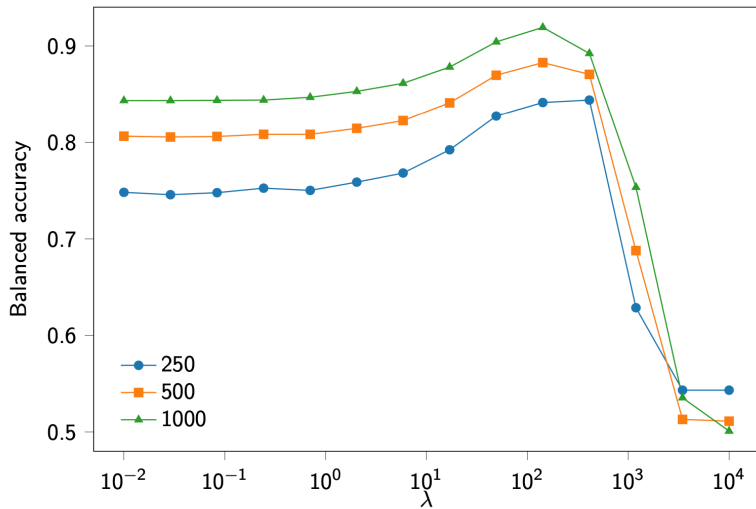
▶ Daily equivalent of a model Hardy (2001) estimated from monthly stock returns

Balanced accuracy (BAC)

True/Predicted	State 1	State 2	Accuracy
State 1	90	0	100%
State 2	10	0	0%

- Average accuracy: $\frac{90}{100} = 90\%$
- Balanced accuracy: $\frac{100\% + 0\%}{2} = 50\%$

Selecting the jump penalty

Parameter estimates based on 1000 simulations

	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC
True	.002	.012	.997 (.015)	.875 (.234)	.954 (.103)
250					
MLE	.288 (.242)	.371 (.285)	.775 (.203)	**.829** (.235)	.752 (.188)
Spec	.100 (.054)	.161 (.059)	.718 (.187)	.759 (.189)	.692 (.153)
Jump	**.006** (.006)	**.024** (.028)	**.861** (.146)	.826 (.240)	**.830** (.146)
500					
MLE	.181 (.231)	.229 (.261)	.865 (.189)	**.865** (.207)	.829 (.185)
Spec	.077 (.056)	.146 (.060)	.791 (.187)	.773 (.190)	.756 (.155)
Jump	**.003** (.002)	**.020** (.021)	**.920** (.124)	.846 (.210)	**.874** (.138)
1000					
MLE	.096 (.180)	.138 (.214)	.937 (.133)	**.885** (.178)	.896 (.145)
Spec	.053 (.046)	.133 (.057)	.852 (.160)	.794 (.163)	.814 (.125)
Jump	**.003** (.002)	**.019** (.016)	**.967** (.073)	.873 (.151)	**.917** (.094)

Robustness to misspecification

	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC
Conditional Gaussian distributions					
True	.002	.012	.997 (.015)	.875 (.234)	.954 (.103)
MLE	.181 (.231)	.229 (.261)	.865 (.189)	**.865** (.207)	.829 (.185)
Spec	.077 (.056)	.146 (.060)	.791 (.187)	.773 (.190)	.756 (.155)
Jump	**.003** (.002)	**.020** (.021)	**.920** (.124)	.846 (.210)	**.874** (.138)
Conditional t_5-distributions					
True	.002	.012	.987 (.023)	.824 (.275)	.929 (.123)
MLE	.118 (.134)	.415 (.349)	.926 (.110)	.694 (.293)	.837 (.146)
Spec	.057 (.040)	.147 (.065)	.802 (.163)	.689 (.242)	.748 (.135)
Jump	**.004** (.011)	**.039** (.043)	**.937** (.122)	**.743** (.294)	**.859** (.150)
Negative binomial sojourn-time distributions					
True	.002*	.012*	.974 (.137)	.833 (.327)	.938 (.145)
MLE	.309 (.242)	.345 (.253)	.768 (.217)	.788 (.255)	.740 (.200)
Spec	.107 (.058)	.152 (.056)	.719 (.226)	.744 (.219)	.694 (.176)
Jump	**.005** (.009)	**.019** (.023)	**.866** (.186)	**.843** (.251)	**.842** (.165)

Online parameter estimates

	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC
True	.002	.012	.970 (.074)	.801 (.277)	.937 (.115)
250					
MLE	.246 (.235)	.491 (.303)	.701 (.233)	**.710** (.241)	.718 (.186)
Jump	**.015** (.036)	**.068** (.100)	**.842** (.180)	.664 (.346)	**.802** (.178)
500					
MLE	.149 (.212)	.385 (.323)	.792 (.230)	**.778** (.248)	.806 (.194)
Jump	**.011** (.018)	**.061** (.091)	**.898** (.163)	.712 (.321)	**.859** (.168)
1000					
MLE	.069 (.139)	.274 (.312)	.856 (.213)	**.841** (.231)	.868 (.163)
Jump	**.007** (.013)	**.061** (.113)	**.949** (.099)	.722 (.300)	**.895** (.143)

Online parameters estimates with intraday data

	γ_{12}	γ_{21}	Accuracy 1	Accuracy 2	BAC
True	.002	.012	.970 (.074)	.801 (.277)	.937 (.115)
1	.011 (.018)	.061 (.091)	.898 (.163)	.712 (.321)	.859 (.168)
2	.011 (.036)	.056 (.100)	.902 (.157)	.751 (.333)	.868 (.169)
5	.009 (.020)	.046 (.086)	.903 (.160)	.776 (.332)	.877 (.170)
10	.008 (.017)	.053 (.115)	.907 (.159)	.799 (.313)	.886 (.159)

Each series consists of 500 observations for in-sample estimation and 250 observations for out-of-sample testing. The leftmost column shows the daily sampling frequency used in estimating the standard deviation features.
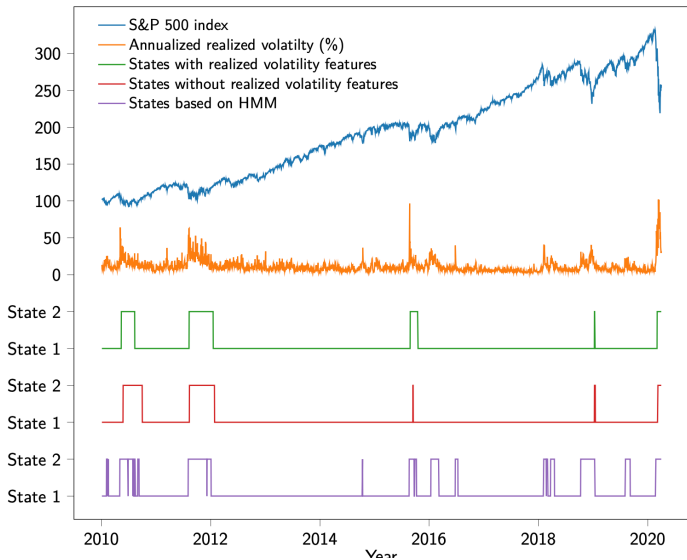
Application to the S&P 500

Let us return to the S&P 500

So what is the state sequence?
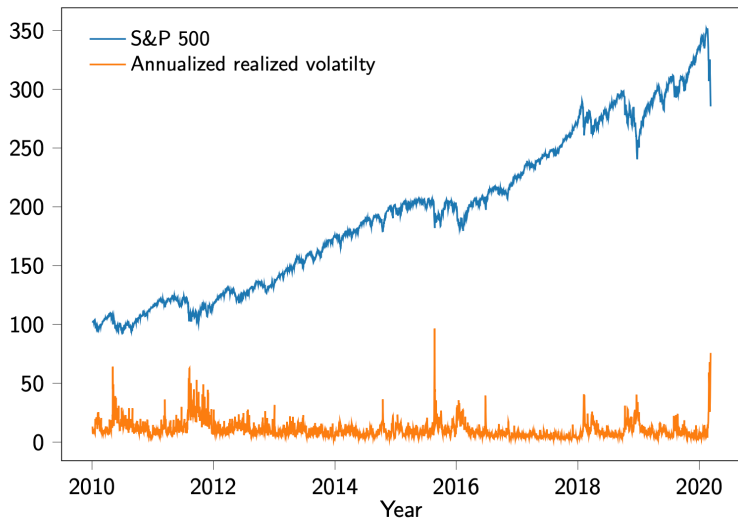
S&P 500 state sequence estimation

Conclusions

Main take-aways about the jump estimator for state classification:

- ▶ Learns hidden state sequence and model parameters simultaneously
- ▶ Provides control over transition rate
- ▶ Converges quickly and is less sensitive to initial values
- ▶ Delivers more accurate estimates of transition probabilities and states
- ▶ Is more robust to misspecification than alternatives
- ▶ Its feature space can be easily extended, thereby improving accuracy

Extensions

Machine learning:

- ▶ Including additional (exogenous) features and feature selection (submitted paper; Nystrup, Kolm, and Lindström (2021))
- ▶ Fading memory and time-varying parameters
- ▶ Application of other loss functions

Financial applications:

- ▶ Strategic asset allocation with regimes
- ▶ Portfolio construction with regimes
- ▶ Regime and state identification in limit order books

References

Bemporad, Alberto et al. (2018). "Fitting jump models". In: *Automatica* 96, pp. 11–21.

Hardy, Mary R. (2001). "A Regime-Switching Model of Long-Term Stock Returns". In: *North American Actuarial Journal* 5.2, pp. 41–53.

Nystrup, Peter, Petter N. Kolm, and Erik Lindström (2020). "Greedy Online Classification of Persistent Market States Using Realized Intraday Volatility Features". In: *Journal of Financial Data Science* 2.3, pp. 25–39.

— (2021). "Feature Selection in Jump Model". In: *submitted paper.*