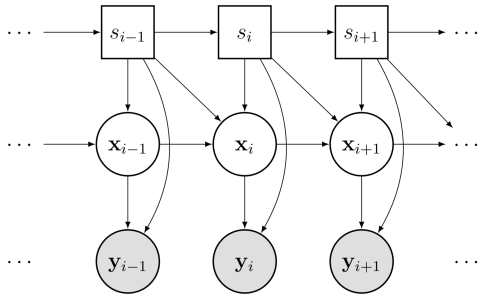
Short overview of `dpf`

Michael McBride

2018-12-14

This package is intended to estimate Markov switching state-space models using maximum likelihood *fast*. The basic model this package examines is illustrated with the graphical model:



where filled objects are observed, rectangles are discrete, and circles are continuous. Switching state-space models have a long history with many applications from economics (Kim and Nelson 1998; Kim 1994; Hamilton 2011) to speech processing (Fox et al. 2011) to animal movement (Patterson et al. 2008; Block et al. 2011). An excellent overview of the history, typography, and algorithmic developments can be found in (Ghahramani and Hinton 2000).

Our implementation of the Markov switching model allows the parameter matrices of a linear Gaussian state-space model to depend on s_i, s_{i-1}:

$$\begin{aligned} x_1 &\sim N(x_0,\ P_0), \\ x_{i+1} &= d(s_i, s_{i-1}) + T(s_i, s_{i-1})x_i + R(s_i, s_{i-1})\eta_i, \quad \eta_i \sim N(0, Q(s_i, s_{i-1})), \\ y_i &= c(s_i) + Z(s_i)x_i + \epsilon_i, \qquad\qquad\qquad\quad \epsilon_i \sim N(0, G(s_i)). \end{aligned} \tag{1}$$

In other words, the hidden Markov (switch) state determines which parameter matrices govern the evolution of the system. If the matrices above depend on a vector of unknown parameters θ, one needs to jointly estimate the switch path $S_1, \ldots, S_n$, the continuous hidden states $X_1, \ldots, X_n$, and the parameter vector.

Why this package?

Most available packages on CRAN are for state-space models only (not switching) or for hidden Markov models only (without continuous hidden states). Without switch states, the Kalman filter (Kalman 1960) returns the likelihood of a parameter vector θ, and the Kalman smoother gives the estimates of the hidden continuous states conditional on all the observed data. In the case of an HMM, the Viterbi algorithm is analogous to the Kalman smoother, producing the most likely path conditional on all the data.

However, when combined, computing the likelihood for θ or producing the most likely path becomes exponentially hard: given 2 possible values for S_i, there are 2^n possible sequences $(S_1, \ldots, S_n)$. Maximizing the likelihood for θ then means evaluating the Kalman filter at each of these paths *for each proposed value of* θ.

Contribution

The `dpf` package provides a lightweight, fast, greedy method for calculating the likelihood of θ. It is implemented in C++ using `Rcpp` and takes advantage of an accurate algorithm for finding the most likely

state sequence proposed in (Fearnhead and Clifford 2003).

We provide 3 main functions which are useful for generic switching state space models:

1. `kalman()` computes the standard Kalman filter and smoother given a collection of parameter matrices which are allowed to be time dependent. It returns the likelihood, means and variances for the filter and smoother distributions, and predicted values for the observations.
2. `beamSearch()` implements the Greedy HMM algorithm. For any N, it attempts to compute the N most likely sequences $(S_1, \ldots, S_n)$ given a collection of parameter matrices which are time and state dependent. The result is this collection of paths as well as probabilities associated with each of the N paths. Using the path with the highest weight in `kalman()` then corresponds to a frequentist evaluation of a particular parameter θ while sampling proportional to the weights allows for approximate Bayesian inference (ignoring the $s^N - N$ paths which presumably have negligible likelihood).
3. `getLogLike()` computes only the components of the Kalman filter necessary to evaluate the likelihood.

For a generic switching state-space model whose evolution matrices depend on an unknown vector θ, one would use `beamSearch()` and `getLogLike()` combined with any standard optimization method to produce an ML estimate $\widehat{\theta}$. Then, for $\widehat{\theta}$, one would use `beamSearch()` and `kalman()` to return the ML estimates for $(S_1, \ldots, S_n)$ and $(X_1, \ldots, X_n)$.

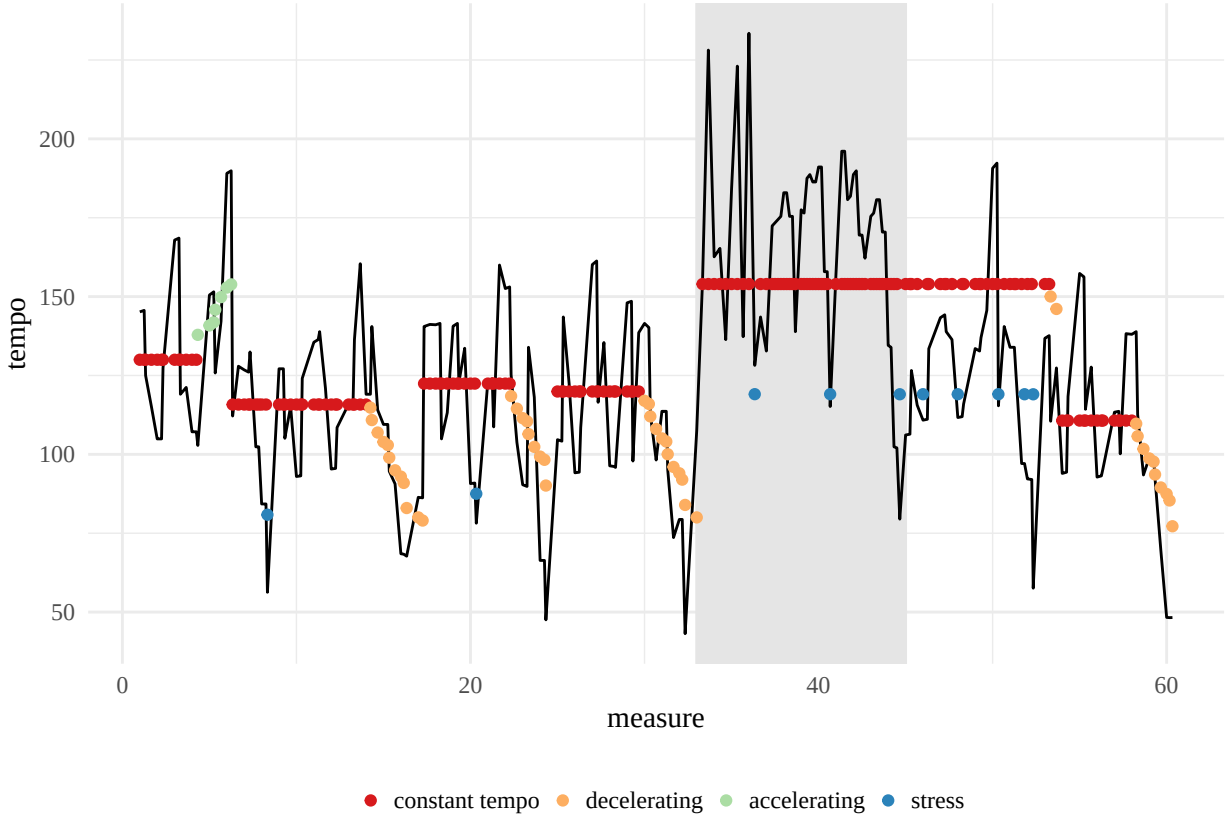
The next section provides a brief illustration of applying these functions to an application from classical music.

Switching models for musical tempo decisions

The remaining functions in `dpf-package` are created for our in-progress manuscript which proposes a switching model for the decisions made by classical music performers. Our model essentially imagines that musicians have an intentional tempo for a piece of music which is partially determined by a Markov chain on 4 states: constant tempo, acceleration, deceleration, and stress. While our working manuscript provides the details, we here illustrate the `beamSearch()` function on a recording of Chopin's Mazurka Op. 68 No. 3 made by the famous Russian pianist Nina Milkina in 1970. We estimated θ offline. The `musicModel()` function creates the appropriate parameter matrices in the above equation given θ.

```
y = matrix(tempos[,'Milkina_1970'], 1)
pmats = musicModel(lt, theta[1], theta[2:4], theta[5:7], theta[8:14],
                   c(132,0), c(400,10)) # prior means and variances on X_1
beam = with(pmats, beamSearch(a0, P0, c(1,0,0,0,0,0,0,0,0,0), dt, ct, Tt, Zt,
                  HHt, GGt, y, transMat, 200))
bestpath = with(beam, paths[which.max(weights),])
kal = kalman(pmats, bestpath, y)
```

This figure shows the observed note-by-note speed of the recording (black line) and the estimated discrete states (colored dots). The musical form is in three sections (ABA). The contrasting "B" section is marked with a grey area.



Installation

Using the `devtools` package, one can install the most current version of `dpf` with

```
devtools::install_github('dajmcdon/dpf')
```

References

Block, Barbara A, Ian D Jonsen, Salvador J Jorgensen, Arliss J Winship, Scott A Shaffer, Steven J Bograd, Elliott Lee Hazen, et al. 2011. "Tracking Apex Marine Predator Movements in a Dynamic Ocean." *Nature* 475 (7354): 86.

Fearnhead, Paul, and Peter Clifford. 2003. "On-Line Inference for Hidden Markov Models via Particle Filters." *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 65 (4): 887–99.

Fox, Emily B., Erik B. Sudderth, Michael I. Jordan, and Alan S. Willsky. 2011. "A Sticky HDP-HMM with Application to Speaker Diarization." *The Annals of Applied Statistics* 5 (2A): 1020–56.

Ghahramani, Zoubin, and Geoffrey E Hinton. 2000. "Variational Learning for Switching State-Space Models." *Neural Computation* 12 (4): 831–64.

Hamilton, J.D. 2011. "Calling Recessions in Real Time." *International Journal of Forecasting* 27: 1006–1126.

Kalman, Rudolf E. 1960. "A New Approach to Linear Filtering and Prediction Problems." *Journal of Basic Engineering* 82 (1): 35–45.

Kim, Chang-Jin. 1994. "Dynamic Linear Models with Markov-Switching." *Journal of Econometrics* 60 (1-2): 1–22.

Kim, C.J., and C.R. Nelson. 1998. "Business Cycle Turning Points, a New Coincident Index, and Tests of Duration Dependence Based on a Dynamic Factor Model with Regime Switching." *Review of Economics and Statistics* 80 (2): 188–201.

Patterson, Toby A, Len Thomas, Chris Wilcox, Otso Ovaskainen, and Jason Matthiopoulos. 2008. "State–Space Models of Individual Animal Movement." *Trends in Ecology & Evolution* 23 (2): 87–94.