State space models

Jan-Ole Koslik

This vignette shows how to fit state space models which can be interpreted as generalization of HMMs to continuous state spaces. Several approaches exist to fitting such models, but Langrock (2011) showed that very general state space models can be fitted via approximate maximum likelihood estimation, when the continous state space is finely discretized. Here, we will showcase this approach for a basic stochastic volatily model, which can be used to describe fincancial markets. In this model the unobserved marked volatility is described by an AR(1) process:

$$g_t = \phi g_{t-1} + \sigma \eta_t, \qquad \eta_t \sim N(0,1),$$

with autoregression parameter $\phi < 1$ a dispersion parameter σ. We could then model share returns y_t as

$$y_t = \beta \epsilon_t \exp(g_t/2),$$

where $\epsilon_t \sim N(0,1)$ and $\beta > 0$ is the baseline standard deviation of the returns (when g_t is in equilibrium), which implies

$$y_t \mid g_t \sim N(0, (\beta e^{g_t/2})^2).$$

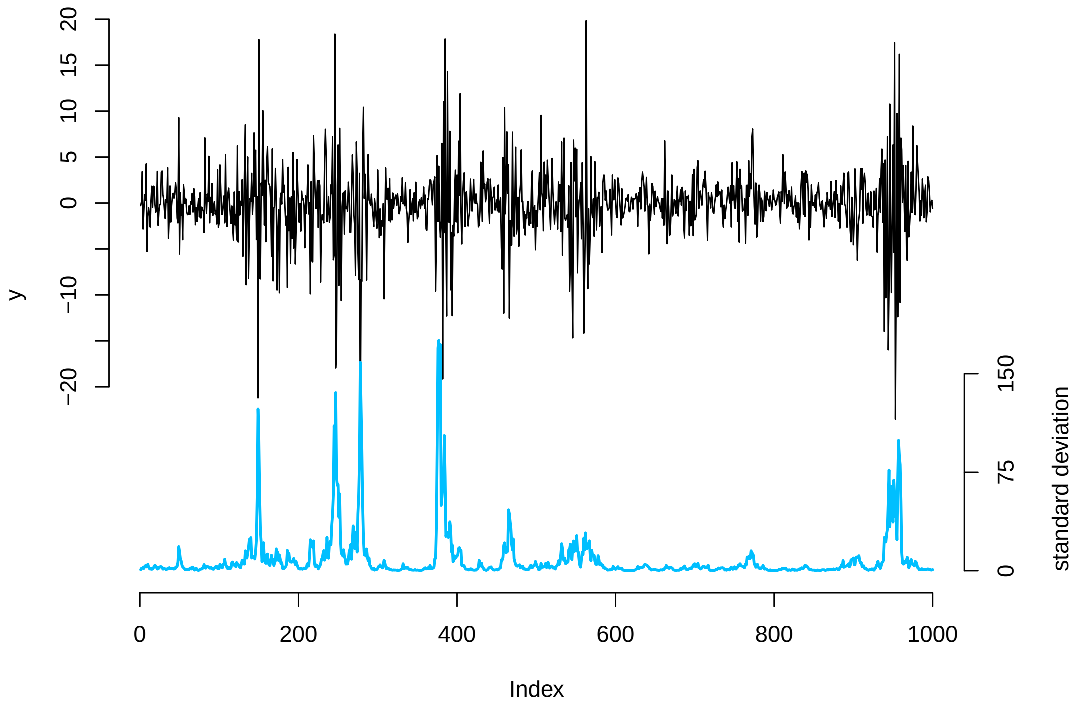
Simulating data from the stochastic volatility model

We start by simulating data from the above specified model:

```r
beta = 2 # baseline standard deviation
phi = 0.95 # AR parameter
sigma = 0.5 # variability of the AR process

n = 1000
set.seed(123)
g = y = rep(NA, n)
g[1] = rnorm(1, 0, sigma / sqrt(1-phi^2)) # stationary distribution of AR process
y[1] = stats::rnorm(1, 0, beta*exp(g[1]/2))
# conditional distribution of y_1 given underlying volatility
for(t in 2:n){
  g[t] = rnorm(1, phi*g[t-1] , sigma) # transition density
  y[t] = stats::rnorm(1, 0, beta*exp(g[t]/2))
  # conditional distribution of y_t given underlying volatility
}

# share returns
par(mar = c(5,4,3,4.5)+0.1)
plot(y, type = "l", bty = "n", ylim = c(-40,20), yaxt = "n")
# true underlying standard deviation
lines(beta*exp(g)/7 - 40, col = "deepskyblue", lwd = 2)
axis(side=2, at = seq(-20,20,by=5), labels = seq(-20,20,by=5))
axis(side=4, at = seq(0,150,by=75)/7-40, labels = seq(0,150,by=75))
mtext("standard deviation", side=4, line=3, at = -30)
```



Writing the negative log-likelihood function

This likelihood formulation corresponds to a fine discretization of the continuous state space into the intervals `b` with width `h` and midpoints `bstar`.

```
mllk = function(theta.star, y, bm, m){
  phi = plogis(theta.star[1])
  sigma = exp(theta.star[2])
  beta = exp(theta.star[3])
  b = seq(-bm, bm, length = m+1) # intervals for midpoint quadrature
  h = b[2]-b[1] # interval width
  bstar = (b[-1] + b[-(m+1)])/2 # interval midpoints
  # approximation resulting from midpoint quadrature
  Gamma = sapply(bstar, dnorm, mean = phi*bstar, sd = sigma) * h
  Gamma = Gamma / rowSums(Gamma) # normalizing out approximation errors
  delta = h * dnorm(bstar, 0, sigma/sqrt(1-phi^2)) # stationary distribution
  # approximating state-dependent density based on midpoints
  allprobs = t(sapply(y, dnorm, mean = 0, sd = beta * exp(bstar/2)))
  # return negative for minimization
  -forward(delta, Gamma, allprobs)
}
```

Fitting an SSM to the data

```
theta.star = c(qlogis(0.95), log(0.3), log(1))
bm = 5 # relevant range of underlying volatility (-5,5)
m = 50 # number of approximating states
```

```r
t1 = Sys.time()
mod = stats::nlm(mllk, theta.star, y = y, bm = bm, m = m)
Sys.time()-t1
#> Time difference of 0.201272 secs
```

Results

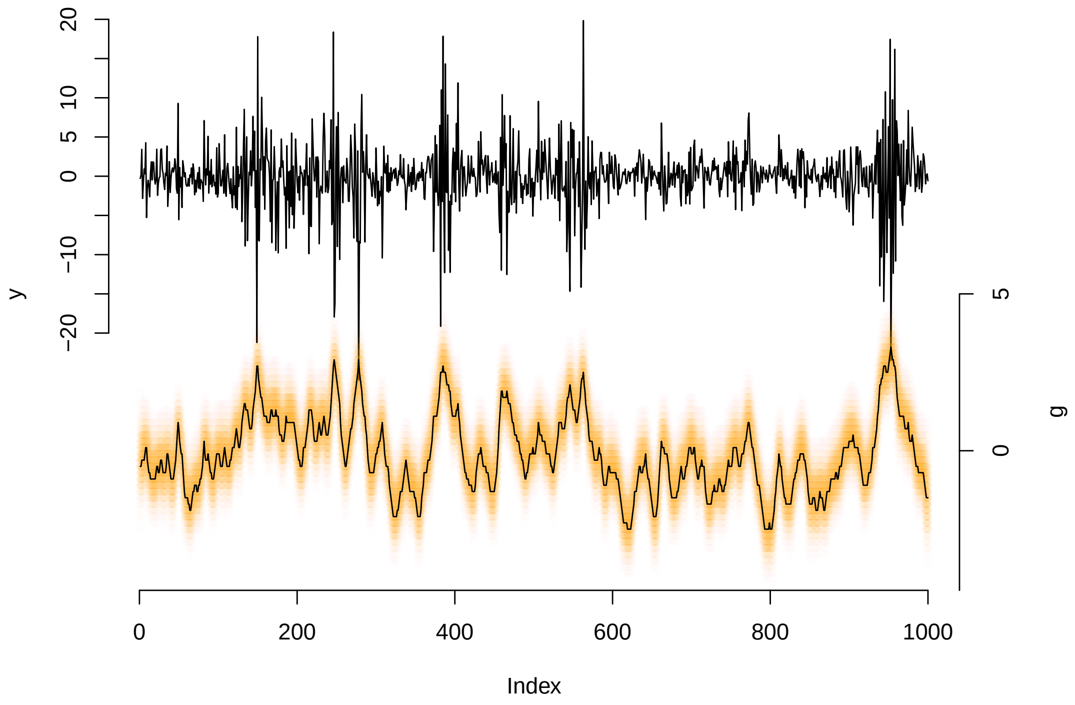
```r
# parameter estimates
(phi = plogis(mod$estimate[1]))
#> [1] 0.9305141
(sigma = exp(mod$estimate[2]))
#> [1] 0.4805211
(beta = exp(mod$estimate[3]))
#> [1] 2.50383

# decoding states
b = seq(-bm, bm, length = m+1) # intervals for midpoint quadrature
h = b[2]-b[1] # interval width
bstar = (b[-1] + b[-(m+1)])/2 # interval midpoints
Gamma = sapply(bstar, dnorm, mean = phi*bstar, sd = sigma) * h
Gamma = Gamma / rowSums(Gamma) # normalizing out approximation errors
delta = h * dnorm(bstar, 0, sigma/sqrt(1-phi^2)) # stationary distribution
# approximating state-dependent density based on midpoints
allprobs = t(sapply(y, dnorm, mean = 0, sd = beta * exp(bstar/2)))

probs = stateprobs(delta, Gamma, allprobs)
states = viterbi(delta, Gamma, allprobs)

par(mar = c(5,4,3,4.5)+0.1)
plot(y, type = "l", bty = "n", ylim = c(-50,20), yaxt = "n")
# when there are so many states it is not too sensable to only plot the most probable state,
# as its probability might still be very small. Generally, we are approximating continuous
# distributions, thus it makes sense to plot the entire conditional distribution.
maxprobs = apply(probs, 1, max)
for(t in 1:1000){
  colend = round((probs[t,]/(maxprobs[t]*5))*100)
  colend[which(colend<10)] = paste0("0", colend[which(colend<10)])
  points(rep(t, m), bstar*4-35, col = paste0("#FFA200",colend), pch = 20)
}
# we can add the viterbi decoded volatility levels as a "mean"
lines(bstar[states]*4-35)

axis(side=2, at = seq(-20,20,by=5), labels = seq(-20,20,by=5))
axis(side=4, at = seq(-5,5, by = 5)*4-35, labels = seq(-5,5, by = 5))
mtext("g", side=4, line=3, at = -30)
```



References

Langrock, Roland. 2011. "On Some Special-Purpose Hidden Markov Models." PhD thesis, University of Göttingen.