06_stochastic_volatility

September 29, 2021

1 Stochastic Volatility model

[1]:
```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pymc3 as pm
from pymc3.distributions.timeseries import GaussianRandomWalk

from scipy import optimize
```

Asset prices have time-varying volatility (variance of day over day `returns`). In some periods, returns are highly variable, while in others very stable. Stochastic volatility models model this with a latent volatility variable, modeled as a stochastic process. The following model is similar to the one described in the No-U-Turn Sampler paper, Hoffman (2011) p21.

$$\sigma \sim Exponential(50)$$

$$\nu \sim Exponential(.1)$$

$$s_i \sim Normal(s_{i-1}, \sigma^{-2})$$

$$log(r_i) \sim t(\nu, 0, exp(-2s_i))$$

Here, r is the daily return series and s is the latent log volatility process.

1.1 Build Model

First we load some daily returns of the S&P 500.

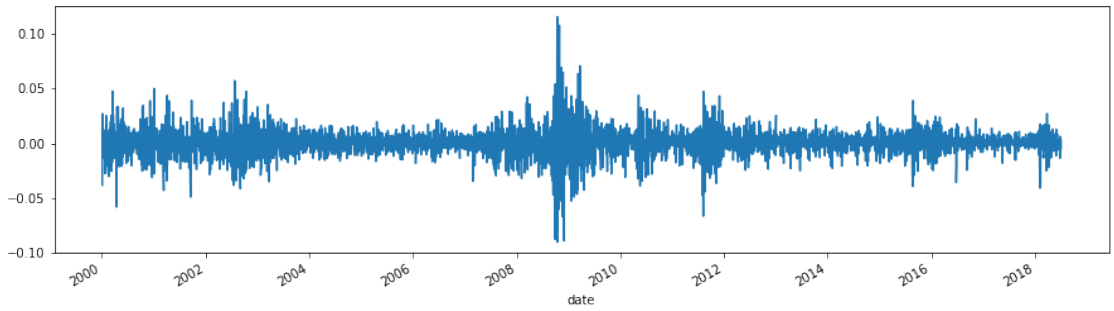
[2]:
```
n = 400
returns = pd.read_hdf('../data/assets.h5', key='sp500/prices').loc['2000':,␣
↪'close'].pct_change().dropna()
returns[:5]
```

[2]:
```
date
2000-01-04   -0.038345
2000-01-05    0.001922
2000-01-06    0.000956
2000-01-07    0.027090
2000-01-10    0.011190
Name: close, dtype: float64
```

As you can see, the volatility seems to change over time quite a bit but cluster around certain time-periods. Around time-points 2500-3000 you can see the 2009 financial crash.

[3]:
```
returns.plot(figsize=(15,4))
```

[3]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fbb14184eb8>
```



Specifying the model in PyMC3 mirrors its statistical specification.

[4]:
```
with pm.Model() as model:
    step_size = pm.Exponential('sigma', 50.)
    s = GaussianRandomWalk('s', sd=step_size, shape=len(returns))

    nu = pm.Exponential('nu', .1)
    r = pm.StudentT('r', nu=nu, lam=pm.math.exp(-2*s),
                    observed=returns)
```

```
INFO (theano.gof.compilelock): Waiting for existing lock by process '17767' (I
am process '17791')
INFO (theano.gof.compilelock): To manually release the lock, delete
/home/stefan/.theano/compiledir_Linux-4.15--generic-x86_64-with-debian-buster-
sid-x86_64-3.7.1-64/lock_dir
INFO (theano.gof.compilelock): Waiting for existing lock by process '17767' (I
am process '17791')
INFO (theano.gof.compilelock): To manually release the lock, delete
/home/stefan/.theano/compiledir_Linux-4.15--generic-x86_64-with-debian-buster-
sid-x86_64-3.7.1-64/lock_dir
INFO (theano.gof.compilelock): Waiting for existing lock by process '17767' (I
```

```
am process '17791')
INFO (theano.gof.compilelock): To manually release the lock, delete
/home/stefan/.theano/compiledir_Linux-4.15--generic-x86_64-with-debian-buster-
sid-x86_64-3.7.1-64/lock_dir
INFO (theano.gof.compilelock): Waiting for existing lock by process '17767' (I
am process '17791')
INFO (theano.gof.compilelock): To manually release the lock, delete
/home/stefan/.theano/compiledir_Linux-4.15--generic-x86_64-with-debian-buster-
sid-x86_64-3.7.1-64/lock_dir
INFO (theano.gof.compilelock): Waiting for existing lock by process '17767' (I
am process '17791')
INFO (theano.gof.compilelock): To manually release the lock, delete
/home/stefan/.theano/compiledir_Linux-4.15--generic-x86_64-with-debian-buster-
sid-x86_64-3.7.1-64/lock_dir
```

1.2 Fit Model

For this model, the full maximum a posteriori (MAP) point is degenerate and has infinite density. NUTS, however, gives the correct posterior.

[5]:
```
with model:
    trace = pm.sample(tune=2000, nuts_kwargs=dict(target_accept=.9))
```

```
Auto-assigning NUTS sampler…
Initializing NUTS using jitter+adapt_diag…
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [nu, s, sigma]
Sampling 4 chains:  12%|         | 1152/10000 [41:50<7:42:18,  3.14s/draws]
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
~/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-packages/pymc3
 ↪sampling.py in _mp_sample(draws, tune, step, chains, cores, chain,␣
 ↪random_seed, start, progressbar, trace, model, use_mmap, **kwargs)
    998             with sampler:
--> 999                 for draw in sampler:
   1000                     trace = traces[draw.chain - chain]

~/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-packages/pymc3
 ↪parallel_sampling.py in __iter__(self)
    304         while self._active:
--> 305             draw = ProcessAdapter.recv_draw(self._active)
    306             proc, is_last, draw, tuning, stats, warns = draw

~/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-packages/pymc3
 ↪parallel_sampling.py in recv_draw(processes, timeout)
    213         pipes = [proc._msg_pipe for proc in processes]
--> 214         ready = multiprocessing.connection.wait(pipes)
```

```
    215         if not ready:

~/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/multiprocessing/
 ↪connection.py in wait(object_list, timeout)
    919             while True:
--> 920                 ready = selector.select(timeout)
    921                 if ready:

~/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/selectors.py in␣
 ↪select(self, timeout)
    414         try:
--> 415             fd_event_list = self._selector.poll(timeout)
    416         except InterruptedError:

KeyboardInterrupt:

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
<ipython-input-5-447f35627412> in <module>
      1 with model:
----> 2     trace = pm.sample(tune=2000, nuts_kwargs=dict(target_accept=.9))

~/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-packages/pymc3
 ↪sampling.py in sample(draws, step, init, n_init, start, trace, chain_idx,␣
 ↪chains, cores, tune, nuts_kwargs, step_kwargs, progressbar, model,␣
 ↪random_seed, live_plot, discard_tuned_samples, live_plot_kwargs,␣
 ↪compute_convergence_checks, use_mmap, **kwargs)
    447             _print_step_hierarchy(step)
    448             try:
--> 449                 trace = _mp_sample(**sample_args)
    450             except pickle.PickleError:
    451                 _log.warning("Could not pickle model, sampling␣
 ↪singlethreaded.")

~/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-packages/pymc3
 ↪sampling.py in _mp_sample(draws, tune, step, chains, cores, chain,␣
 ↪random_seed, start, progressbar, trace, model, use_mmap, **kwargs)
   1009             return MultiTrace(traces)
   1010         except KeyboardInterrupt:
-> 1011             traces, length = _choose_chains(traces, tune)
   1012             return MultiTrace(traces)[:length]
   1013         finally:

~/.pyenv/versions/miniconda3-latest/envs/ml4t/lib/python3.7/site-packages/pymc3
 ↪sampling.py in _choose_chains(traces, tune)
   1042     lengths = [max(0, len(trace) - tune) for trace in traces]
   1043     if not sum(lengths):
-> 1044         raise ValueError('Not enough samples to build a trace.')
```

```
   1045
   1046     idxs = np.argsort(lengths)[::-1]

ValueError: Not enough samples to build a trace.
```

[ ]:
```
pm.traceplot(trace, varnames=['sigma', 'nu']);
```

[ ]:
```
fig, ax = plt.subplots()

plt.plot(trace['s'].T, 'b', alpha=.03);
ax.set(title=str(s), xlabel='time', ylabel='log volatility');
```

Looking at the returns over time and overlaying the estimated standard deviation we can see how the model tracks the volatility over time.

[ ]:
```
pm.trace_to_dataframe(trace).info()
```

[ ]:
```
fig, ax = plt.subplots(figsize=(14, 8))
ax.plot(returns.values)
ax.plot(np.exp(trace[s]).T, 'r', alpha=.03);
ax.set(xlabel='time', ylabel='returns')
ax.legend(['S&P500', 'stoch vol']);
```