mcmc_sampler

May 29, 2019

```python
In [8]: import numpy as np
        import emcee
        from tqdm import tqdm
        import posterior as pos

        posterior = pos.posterior()

In [2]: lmax = 500

        ns_range = [0.85, 1.15, 0.005]
        file_scal = 'cls_scal_lmax2500_ns0p85-1p15_step0p005.npy'
        file_tens = 'cls_tens_lmax2500.npy'
        file_data = 'cls_data_lmax2500.npy'

        posterior.load_theory(file_scal, file_tens, lmax, ns_range)
        posterior.load_data(file_data, lmax)

In [3]: filename = 'chains/test.chains'
        name = '000'

        ndim = 3
        m0   = [1., 1., 0.]
        sig0 = [0.001, 0.01, 0.1]

        nwalkers = 10
        max_n = 10000

In [4]: backend = emcee.backends.HDFBackend(filename, name=name)
        #backend.reset(nwalkers, ndim)

        sampler = emcee.EnsembleSampler(nwalkers, ndim, posterior.lnpos,
                                        args=[[ns_range[0], ns_range[1]],[]], backend=backend)

In [6]: if sampler.iteration == 0:
            p0 = np.zeros((nwalkers, ndim))
            m0 = np.array(m0)
            sig0 = np.array(sig0)
            for i in range(nwalkers):
```

```python
            p0 [i,:] = np.random.normal(size=ndim)*sig0 + m0
            while posterior.lnprior(p0[i,:], [ns_range[0], ns_range[1]]) == -np.inf:
                p0[i,:] = np.random.normal(size=ndim)*sig0 + m0
    else:
        print 'Backend file already exists! Continuing with the sampling from last point'
        p0 = sampler.chain[:,-1,:]
```

In [7]:
```python
old_tau = np.inf

ite = tqdm(sampler.sample(p0, iterations=max_n), total=max_n)
for sample in ite:
    if sampler.iteration % 100:
        continue

    tau = sampler.get_autocorr_time(tol=0)
    af = sampler.acceptance_fraction

    ite.set_description('ind: '+str(int(round(float(sampler.iteration)/max(tau))))
                        +' | tau: '+str([int(round(t)) for t in tau])+' | rat: '
                        +str(round(min(af),2))+'-'+str(round(max(af),2))+' ')

    # Check convergence
    converged = np.all(tau * 100 < sampler.iteration)
    converged &= np.all(np.abs(old_tau - tau) / tau < 0.01)
    if converged:
        print ' Converged!'
        break
    old_tau = tau
```

```
ind: 105 | tau: [49, 50, 54] | rat: 0.59-0.62 :  57%|     | 5699/10000 [1:18:22<1:10:29,  1.02it/

 Converged!
```