emcee example 2

sampling a multivariate Gaussian using emcee

12/1/16

lnprob

1) Define the probability distribution that you would like to sample.

Below are two examples of code for... a Gaussian dist.?

```
def lnprob(x, mu, icov):
    diff = x-mu
    return -np.dot(diff,np.dot(icov,diff))/2.0
```

```
def lnprob(p, vec):
    diff = vec-p[0]
    N = len(vec)
    return -0.5 * N * np.log(2 * np.pi) - N * np.log(p[1]) - 0.5 \
                                    * np.sum(( (vec - p[0]) / p[1] ) ** 2)
```

SIDE NOTE: From your notes…(week 4, day 3)

```
def lnprob(x):
    if x[2] < 0:
        return -np.infty
    else:
        return -0.5*x[2]*np.sum([(e[1] - x[0] - x[1]*e[0])**2 for e in data]) + 0.5*N*np.log(x[2])
 - 0.5*(x[0]**2+ x[1]**2) - x[2]
```

SIDE NOTE: From your notes…(week 4, day 3)

```python
def lnprob(x):
    if x[2] < 0:
        return -np.infty
    else:
        return -0.5*x[2]*np.sum([(e[1] - x[0] - x[1]*e[0])**2 for e in data]) + 0.5*N*np.log(x[2])
- 0.5*(x[0]**2+ x[1]**2) - x[2]
```

Return lnprior(x) + lnlike(x)

The big difference between `emcee` and `PyStan` and `pymc` is that the module is all about the sampler and doesn't give you any build-in distributions. You have to write the entire probability function yourself. Following the example on `emcee`'s site, we do this by writing log-prior, log-likelihood, and log-probability functions:

In [65]:
```
def lnprior(p):
    # The parameters are stored as a vector of values, so unpack them
    alpha,betax,betay,eps = p
    # We're using only uniform priors, and only eps has a lower bound
    if eps <= 0:
        return -inf
    return 0

def lnlike(p, x, y, z):
    alpha,betax,betay,eps = p
    model = alpha + betax*x + betay*y
    # the likelihood is sum of the lot of normal distributions
    denom = power(eps,2)
    lp = -0.5*sum(power((z - model),2)/denom + log(denom) + log(2*pi))
    return lp

def lnprob(p, x, y, z):
    lp = lnprior(p)
    if not isfinite(lp):
        return -inf
    return lp + lnlike(p, x, y, z)
```

Comparing two different examples for the next steps...

```python
# We'll sample a 10-dimensional Gaussian...
ndim = 10
# ...with randomly chosen mean position...
means = np.random.rand(ndim)
# ...and a positive definite, non-trivial covariance matrix.
cov  = 0.5-np.random.rand(ndim**2).reshape((ndim, ndim))
cov  = np.triu(cov)
cov += cov.T - np.diag(cov.diagonal())
cov  = np.dot(cov,cov)

# Invert the covariance matrix first.
icov = np.linalg.inv(cov)

# We'll sample with 50 walkers.
nwalkers = 50
```

```python
# We'll sample a Gaussian which has 2 parameters: mean and sigma...
ndim = 2

# We'll sample with 250 walkers. (nwalkers must be an even number)
nwalkers = 250
```

2) Choose an initial set of positions for the walkers.

```
# Choose an initial set of positions for the walkers.
p0 = [np.random.rand(ndim) for i in xrange(nwalkers)]
```


```
# Choose an initial set of positions for the walkers.
p0 = [np.random.rand(ndim) for i in xrange(nwalkers)]
```

SIDE NOTE: From your notes…(week 4, day 2)

In [4]:
```
def lnprob(l):
    if l < 0:
        return -np.infty
    else:
        return (sumk + 5-1)*np.log(l) - (N+1)*l

nwalkers = 20
ndim = 1
p0 = np.random.rand(nwalkers*ndim).reshape((nwalkers,ndim))
sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob)
pos, prob, state = sampler.run_mcmc(p0, 1000)
sampler.reset()
pos, prob, state = sampler.run_mcmc(pos, 100000)
samples = sampler.flatchain
```

3) Initialize the sampler with the chosen specs.

```
# Initialize the sampler with the chosen specs.
sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob, args=[means, icov])
```

```
# Initialize the sampler with the chosen specs.
#The "a" parameter controls the step size, the default is a=2,
#but in this case works better with a=4 see below or page 10 in the paper
sampler = emcee.EnsembleSampler(nwalkers, ndim, lnprob, args=[data], a=4)
```

4) Run n steps as a burn in.

```
# Run 5000 steps as a burn-in.
pos, prob, state = sampler.run_mcmc(p0, 5000)
```

```
# Run 200 steps as a burn-in.
print "Burning in ..."
pos, prob, state = sampler.run_mcmc(p0, 200)
```

5) Reset the chain to remove the burn-in samples.

```
# Reset the chain to remove the burn-in samples.
sampler.reset()
```



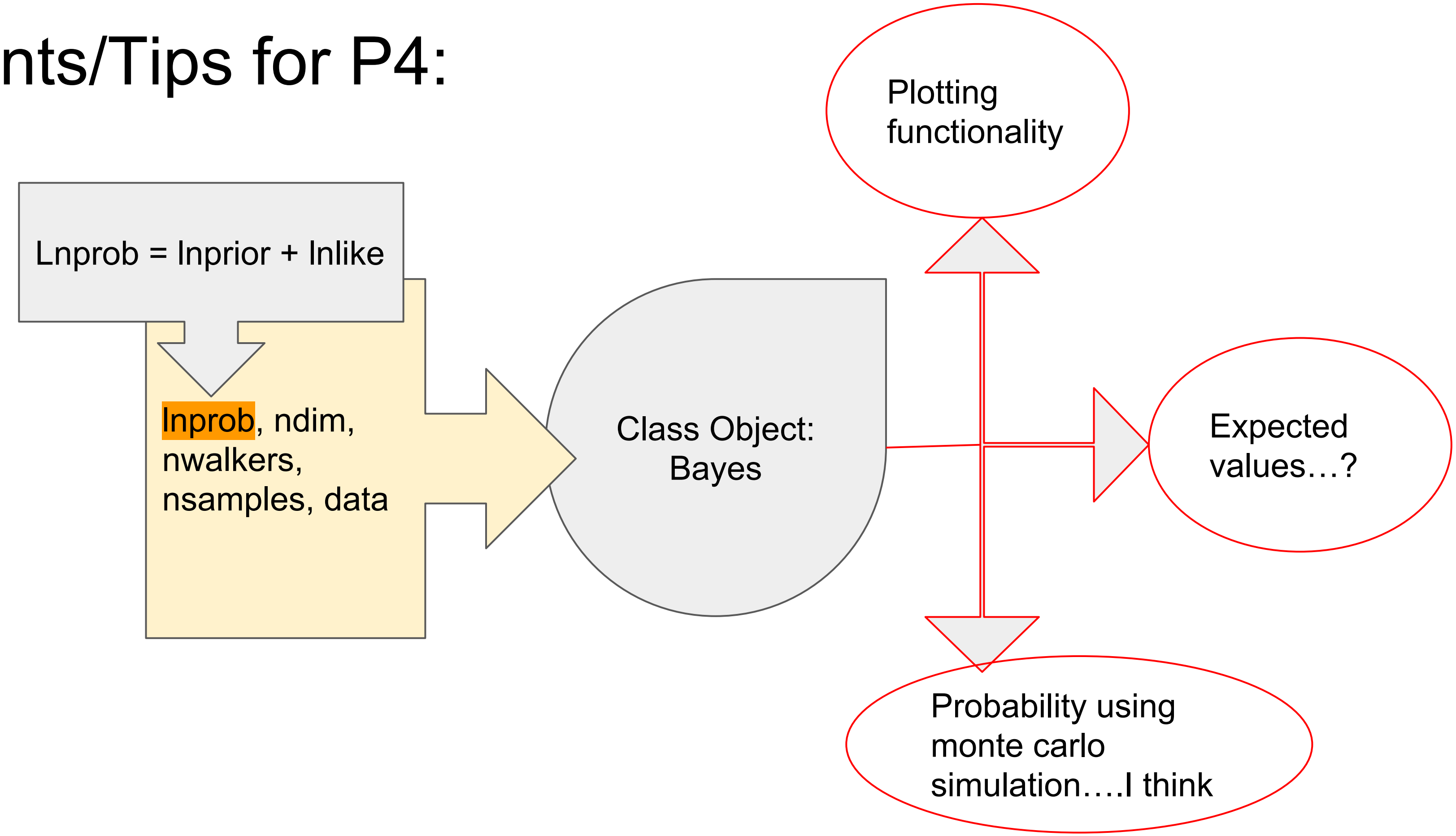
```
# Reset the chain to remove the burn-in samples.
sampler.reset()
```

6) Starting from the final position in the burn-in chain, sample for n steps

```
# Starting from the final position in the burn-in chain, sample for 100000
# steps.
sampler.run_mcmc(pos, 100000, rstate0=state)
```

```
# Starting from the final position in the burn-in chain, sample for 1000
# steps. (rstate0 is the state of the internal random number generator)
print "Running MCMC ..."
pos, prob, state = sampler.run_mcmc(pos, 1000, rstate0=state)
```

Hints/Tips for P4:



Sources

- Emcee gaussian example 1:
  - https://github.com/dfm/emcee/blob/master/examples/quickstart.py
- Emcee gaussian example 2:
  - https://gist.github.com/banados/2254240
- 3rd emcee example:
  - https://users.obs.carnegiescience.edu/cburns/ipynbs/Emcee.html
- Explanation of flatchain:
  - http://dan.iel.fm/emcee/current/api/