Nathan Hadfield

CS 677 (Spring 2014)

# MCMC Lab #2

## Faculty Evaluations

My code for setting up the faculty evaluations model is mostly the same as the sample code, except when it comes to plotting the results. (Code for the nodes and network classes can be found at the end of this report.)

#!/usr/bin/env python

import numpy

from node\_normal import \*

from node\_invgamma import \*

from network import \*

datafilename = 'faculty.dat'

nsamples = 10000

burn = 0

mean\_candsd = 0.2

var\_candsd = 0.15

# Read in Data

data = [float(line) for line in open(datafilename)]

# Use point estimators from the data to come up with starting values.

estimated\_mean = numpy.mean(data)

estimated\_var = numpy.var(data)

def MomentsInvGammaShape(mean, var):

return 1

def MomentsInvGammaScale(mean,var):

return 1

# Create Nodes and Links in Network

meannode = NormalNode(estimated\_mean, name='Mean', cand\_var=mean\_candsd, mean=5, var=(1/3)\*\*2)

varprior\_mean = 1/4

varprior\_stddev = 1/12

varprior\_shape = MomentsInvGammaShape(varprior\_mean, varprior\_stddev\*\*2)

varprior\_scale = MomentsInvGammaScale(varprior\_mean, varprior\_stddev\*\*2)

varnode = InvGammaNode(estimated\_var, name='Variance', cand\_var=var\_candsd, shape=varprior\_shape, scale=varprior\_scale)

for datum in data:

NormalNode(datum, observed=True, mean=meannode, var=varnode)

# Perform simulations and plot results

network = Network([meannode, varnode])

samples = network.collect\_samples(burn, nsamples)

def mean\_prior\_pdf(x):

return stats.norm.pdf(x, 5, 1/3)

def var\_prior\_pdf(x):

return stats.invgamma.pdf(x, a=varprior\_shape, scale=varprior\_scale)

prior\_pdfs = { meannode: mean\_prior\_pdf, varnode: var\_prior\_pdf }

results = {}

for node in [meannode, varnode]:

params = {

'mean': numpy.mean(samples.of\_node(node)),

'var': numpy.var(samples.of\_node(node))

}

results[node] = params

title = "{}: mean = {}, var = {} (burn={}, n={})". \

format(node.pdf\_name, params['mean'], params['var'], burn, nsamples - burn)

samples.plot\_node(node, title=title)

if params['var'] > 0: # histogram fails if all values are the same

samples.plot\_histogram\_for\_node(node, title=title, prior\_pdf=prior\_pdfs[node])

Here are the plots:

|  |  |
| --- | --- |
|  |  |
|  |  |

## Professional Golfers

## Wacky Network

With no observations:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

With G observed to be 5:

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | *(Can’t plot with an empty range.)* |

### network\_wacky.py

from node\_normal import \*

from node\_beta import \*

from node\_gamma import \*

from node\_poisson import \*

from node\_bernoulli import \*

from network import \*

import numpy

logging.basicConfig(level=logging.WARNING,

format='[%(levelname)s] %(module)s %(funcName)s(): %(message)s')

burn = 0

num\_samples = burn + 100000

for g\_observed in [False, True]:

a = NormalNode(20, 'A', mean=20, var=1)

e = BetaNode(0.5, 'E', alpha=1, beta=1)

b = GammaNode(0.2, 'B', shape=a, shape\_modifier=lambda x: x \*\* math.pi, scale=1/7)

d = BetaNode(0.5, 'D', alpha=a, beta=e)

c = BernoulliNode(0, 'C', p=d)

f = PoissonNode(4, 'F', rate=d)

g = NormalNode(5, 'G', mean=e, var=f, observed=g\_observed)

network = Network([a, e, b, d, c, f, g])

samples = network.collect\_samples(burn=burn, n=num\_samples)

for node in network.nodes:

mean = numpy.mean(samples.of\_node(node))

var = numpy.var(samples.of\_node(node))

title = "{} [G observed={}]: mean = {:.4f}, var = {:.4f} (burn={}, n={})"

.format(node.pdf\_name, g\_observed, mean, var, burn, num\_samples-burn)

samples.plot\_node(node, title=title)

samples.plot\_histogram\_for\_node(node, title=title)