Statistical Computing 2

숙제 8

2019년 가을학기

응용통계학과 석사과정 최석준

1. Using Stochastic Approximation Monte Carlo method, generate samples from

$$\frac{1}{3}N(\mathbf{0}, I) + \frac{1}{3}N\left(\begin{bmatrix}-8\\-6\end{bmatrix}, \begin{bmatrix}1 & 0.9\\0.9 & 1\end{bmatrix}\right) + \frac{1}{3}N\left(\begin{bmatrix}8\\6\end{bmatrix}, \begin{bmatrix}1 & -0.9\\-0.9 & 1\end{bmatrix}\right)$$

먼저 Stochastic Approximation Monte Carlo (이하 SAMC)의 로직만을 구현한다.

[Python]

```
#python 3 file created by Choi, Seokjun

#using SAMC,
#get standard normal samples.

from math import log, exp, pi
from random import normalvariate, uniform, seed
import time
from statistics import mean
import os
import multiprocessing as mp
from abc import abstractmethod

import matplotlib.pyplot as plt


class MC_StochasticApproximation:
    def __init__(self, log_target_pdf, partition_indicator, set_visiting_freq, start_const_on_exp_vec, proposal_sigma, initial):
        # code variable name : Lecturnote notation

        self.initial = initial
        self.dim = len(initial)
        self.log_target_pdf = log_target_pdf #function. if input data point, return log-pdf value

        self.partition_indicator = partition_indicator #function. if input data point, then return i (index of partition)
        self.set_visiting_freq = set_visiting_freq #vector. for each partition i. (trivially, 1/n, n=#of partitions)

        self.proposal_sigma = proposal_sigma
        self.normalizing_const_exponent = [tuple(start_const_on_exp_vec)] # vector : theta
        self.MC_sample   = [tuple(initial)]
        self.MC_visiting_idx = [self.partition_indicator(initial)]

        self.num_total_iters = 0
        self.num_accept = 0
        self.pid = None

    def gain_factor(self):
        #gain factor 모양 보며 아래 parameter 2개 조정
        t0 = 2 #should be >1
```

```
        xi = 0.9 #shouwld be 0.5 < xi <= 1

        iter_num = self.num_total_iters
        gain_factor = t0 / max(t0, iter_num**xi)
        return gain_factor

    @abstractmethod
    def proposal_sampler(self, last):
        # 차원에 맞게 multivariate normal d개 생선한 tuple 리턴
        # return normalvariate(last, self.proposal_sigma)
        pass

    def log_proposal_pdf(self, from_smpl, to_smpl):
        #When we calculate log_r(MH ratio's log value), just canceled.
        #since normal distribution is symmetric.
        #so we do not need implement this term.
        return 0

    def log_r_calculator(self, last, last_partition_idx, candid, candid_partition_idx):
        log_r = (self.log_target_pdf(candid) - self.log_proposal_pdf(from_smpl=last, to_smpl=candid) - ₩
                self.log_target_pdf(last) + self.log_proposal_pdf(from_smpl=candid, to_smpl=last))
        now_theta = self.normalizing_const_exponent[-1]
        log_r += now_theta[last_partition_idx] - now_theta[candid_partition_idx]
        return log_r

    def MH_rejection_step(self, last_sample_point, last_partition_idx, candid_sample_point, candid_partition_idx):
        unif_sample = uniform(0, 1)
        if candid_partition_idx is None:
            return False

        try:
            log_r = self.log_r_calculator(last_sample_point, last_partition_idx, candid_sample_point, candid_partition_idx)
        # print(log(unif_sample), log_r) #for debug
        except ValueError:
            return False

        if log(unif_sample) < log_r:
            return True
        else:
            return False

    def Weight_updating_step(self, candid_partition_idx):
        last_theta_vec = self.normalizing_const_exponent[-1]
        now_gain_factor = self.gain_factor()
        new_theta = []
        for i, theta_i in enumerate(last_theta_vec):
            new_theta.append(theta_i - now_gain_factor * self.set_visiting_freq[i])
        new_theta[candid_partition_idx] += now_gain_factor
        return new_theta

    def sampler(self):
        last_sample_point = self.MC_sample[-1]
        last_partition_idx = self.MC_visiting_idx[-1]
        proposal_sample_point = self.proposal_sampler(last_sample_point)
        proposal_partition_idx =self.partition_indicator(proposal_sample_point)
```

```
        #MH step
        accept_bool = self.MH_rejection_step(last_sample_point, last_partition_idx, proposal_sample_point, proposal_partition_idx)

        self.num_total_iters += 1
        if accept_bool:
            self.MC_sample.append(tuple(proposal_sample_point))
            # self.MC_visiting_idx.append(proposal_partition_idx)
            self.num_accept += 1
        else :
            self.MC_sample.append(tuple(last_sample_point))
            # self.MC_visiting_idx.append(last_partition_idx)

        #Weight updating step
        if proposal_partition_idx is not None:
            self.MC_visiting_idx.append(proposal_partition_idx)
            new_theta = self.Weight_updating_step(proposal_partition_idx)
            self.normalizing_const_exponent.append(new_theta)
        else:
            # proposal이 튀어나갔을시 그냥 reject하고, c* 초기화 대신 기존값 사용하게 구현함
            # (c*를 어떻게 잡아야할지...)
            new_theta = self.normalizing_const_exponent[-1]
            self.normalizing_const_exponent.append(new_theta)

    def generate_samples(self, num_samples, verbose=True):
        start_time = time.time()

        for i in range(1, num_samples):
            self.sampler()
            if i%50000 == 0 and verbose and self.pid is not None:
                print("pid:",self.pid," iteration", i, "/", num_samples)
            elif i%50000 == 0 and verbose and self.pid is None:
                print("iteration", i, "/", num_samples)

        elap_time = time.time()-start_time

        if self.pid is not None and verbose:
            print("pid:",self.pid, "iteration", num_samples, "/", num_samples, " done! (elapsed time for execution: ", elap_time//60,"min ", elap_time%60,"sec)")
        elif self.pid is None and verbose:
            print("iteration", num_samples, "/", num_samples, " done! (elapsed time for execution: ", elap_time//60,"min ", elap_time%60,"sec)")

    def burnin(self, num_burn_in):
        self.MC_sample = self.MC_sample[num_burn_in-1:]

    def thinning(self, lag):
        self.MC_sample = self.MC_sample[::lag]
```

gain factor 로는 일반적으로 사용한다고 알려진 꼴을 사용하였다. MC_stochasticApproximation.gain_factor 의 구현을 보라. 또한, weight updating step 에서 parameter bound 를 나갈 시 (즉, proposal point 가 기존 설정한 partition 범위를 나갈 시) 해당 proposal 은 그냥 reject 하고, weight 초기화는 기존 값으로 하도록 구현하였다. 구체적으로는 MC_stochasticApproximation.sampler 의 구현을 보라.

다음으로는 2-dimension case 를 위한 proposal sampler 를 override 하고, 추가로 2 dimension case 에서의 그림 그리는 함수들과 acf 관련 함수, 그리고 SAMC 진단용 visiting frequency histogram 을 구현한다.

[Python]

```
class SAMC_withUtil_2dim(MC_StochasticApproximation):
    def __init__(self, log_target_pdf, partition_indicator, set_visiting_freq, start_const_on_exp_vec, proposal_sigma, initial):
        super().__init__(log_target_pdf, partition_indicator, set_visiting_freq, start_const_on_exp_vec, proposal_sigma, initial)

    #abstractmehtod override
    def proposal_sampler(self, last):
        return (normalvariate(last[0], self.proposal_sigma), normalvariate(last[1], self.proposal_sigma))



    def get_specific_dim_samples(self, dim_idx):
        #dim_idx 0부터 시작함
        if dim_idx >= self.dim:
            raise ValueError("dimension index should be lower than number of dimension. note that index starts at 0")
        return [smpl[dim_idx] for smpl in self.MC_sample]

    def show_scatterplot(self, show=True):
        x_vec = self.get_specific_dim_samples(0)
        y_vec = self.get_specific_dim_samples(1)
        # plt.plot(x_vec, y_vec, '-o', marker=".")
        plt.plot(x_vec, y_vec, 'o', marker=".")
        if show:
            plt.show()

    def get_accept_rate(self):
        try:
            acc_rate = self.num_accept / self.num_total_iters
        except ZeroDivisionError:
            acc_rate = 0
        return acc_rate

    def get_autocorr(self, dim_idx, maxLag):
        y = self.get_specific_dim_samples(dim_idx)
        acf = []
        y_mean = mean(y)
        y = [elem - y_mean   for elem in y]
        n_var = sum([elem**2 for elem in y])
        for k in range(maxLag+1):
            N = len(y)-k
            n_cov_term = 0
            for i in range(N):
                n_cov_term += y[i]*y[i+k]
            try:
                k_lag_acf = n_cov_term / n_var
            except ZeroDivisionError:
                # raise ZeroDivisionError("n_var is too small (underflow raised.)") #전파 필요시
                k_lag_acf = 1
            acf.append(k_lag_acf)
        return acf

    def show_acf(self, dim_idx, maxLag, show=True):
        grid = [i for i in range(maxLag+1)]
```

```
        acf = self.get_autocorr(dim_idx, maxLag)
        plt.ylim([-1,1])
        plt.bar(grid, acf, width=0.3)
        plt.axhline(0, color="black", linewidth=0.8)
        if show:
            plt.show()

    def show_visiting_idx_hist(self, show=True):
        plt.hist(self.MC_visiting_idx, bins=10)
        if show:
            plt.show()

    def get_visiting_idx_count(self, max_idx):
        count_vec = [0 for x in range(max_idx+1)]
        for idx in self.MC_visiting_idx:
            count_vec[idx] += 1
        return count_vec
```

그리고, 이제 우리 문제의 setting 을 위한 요소를 구현한다. mixture 의 log pdf 와 contour plot 그리는 함수를 구현한다. 또한 추가로, SAMC 의 파티션을 구현하는 mixture_partition_indicator 함수를 만든다. 이 보고서에서는 log likelihood (즉 log pdf 값)을 기준으로 주어진 mixture 를 총 10 개의 파티션으로 나눌 것이다.

```
[Python]
#our case
# (1/3)*N((0,0), I) + (1/3)*N((-8,-6), [1, 0.9; 0.9, 1]) + (1/3)*N((8,6), [1, -0.9; -0.9, 1]))
def bivariate_normal_pdf(x_2d, corr, mu_x1, mu_x2, sigmasq_1=1, sigmasq_2=1):
    x1, x2 = x_2d
    ker = (x1-mu_x1)**2/sigmasq_1 + (x2-mu_x2)**2/sigmasq_2 - 2*corr*(x1-mu_x1)*(x2-mu_x2)/(sigmasq_1*sigmasq_2)**0.5
    ker *= (-0.5)/(1-corr**2)
    const = 2*pi*(sigmasq_1*sigmasq_2*(1-corr**2))**0.5
    return exp(ker)/const

def mixture_log_pdf(x_2d):
    try:
        logpdf_val = log(
            bivariate_normal_pdf(x_2d,0,0,0)/3
            + bivariate_normal_pdf(x_2d, 0.9, -8, -6)/3
            + bivariate_normal_pdf(x_2d, -0.9, 8, 6)/3)
    except ValueError:
        err_pdfval = (bivariate_normal_pdf(x_2d,0,0,0)/3
            + bivariate_normal_pdf(x_2d, 0.9, -8, -6)/3
            + bivariate_normal_pdf(x_2d, -0.9, 8, 6)/3)
        err_str = "Underflow: at "+ str(x_2d) + ", pdf value: " + str(err_pdfval) + " is too close to 0."
        raise ValueError(err_str)
    return logpdf_val


def setting_contourplot(start=-10, end=10):
    grid = [x/10 + start for x in range(10*(end-start))]
    mixture_val = [[mixture_log_pdf([x,y]) for x in grid] for y in grid]
    plt.contour(grid, grid, mixture_val, levels=20)


def mixture_partition_indicator(data_point):
    try:
```

```
        logpdfval = mixture_log_pdf(data_point)
    except ValueError:
        return None

    if logpdfval > -3.5:
        return 0
    elif logpdfval > -4.5:
        return 1
    elif logpdfval > -5.6:
        return 2
    elif logpdfval > -6.7:
        return 3
    elif logpdfval > -7.9:
        return 4
    elif logpdfval > -9.3:
        return 5
    elif logpdfval > -10.8:
        return 6
    elif logpdfval > -12.7:
        return 7
    elif logpdfval > -15:
        return 8
    elif logpdfval > -17.5:
        return 9
    else:
        return None
```

이 프로그램을 실행 시, 각 cpu core 가 동시에 8 개 SAMC chain 을 만들도록 병렬 처리할 것이다. 다음은 각 core 가 할 작업을 지시하기 위한 함수이다.

```
[Python]
#for multiprocessing
def multiproc_1unit_do(result_queue, initial, num_iter):
    func_pid = os.getpid()
    print("pid: ", func_pid, "start!")
    SAMCchain = SAMC_withUtil_2dim(log_target_pdf = mixture_log_pdf,
                                   partition_indicator = mixture_partition_indicator,
                                   set_visiting_freq = tuple([1/10 for _ in range(10)]),
                                   start_const_on_exp_vec = tuple([1/10 for _ in range(10)]),
                                   proposal_sigma = 3,
                                   initial = initial
                                   )
    SAMCchain.pid = func_pid
    SAMCchain.generate_samples(num_iter)
    SAMCchain.burnin(100000)
    SAMCchain.thinning(200)

    result_queue.put(SAMCchain)
    print("pid: ", func_pid, " end!")
```

마지막으로, 프로그램의 main 부분이다. 크게 2 부분으로 나뉘어 있다.

첫 부분은 각 cpu core 에 넘길 SAMC setting 과 그 multiprocessing 구현이다. (setting 은 위에 multiproc_1unit 에서도 한다. (세팅 요소를 한군데서 몰아서 하도록 짜는 것이 더 나았을 것 같기도 하다..) 두번째 부분은 동시에 생성된 8 개의 chain 을 가지고 한꺼번에 그림을 그려 출력한다.

[Python]

```
if __name__=="__main__":
    seed(2019311252)
    core_num = 8
    #setting
    initial = [(x,x) for x in range(-4,5)]
    num_iter = 500000



    #generate SAMC chains using parallel multiprocessing
    print("start.mp")
    proc_vec = []
    proc_queue = mp.Queue()

    for i in range(core_num):
        unit_proc = mp.Process(target = multiproc_1unit_do,
            args=(proc_queue, initial[i], num_iter,))
        proc_vec.append(unit_proc)

    for unit_proc in proc_vec:
        unit_proc.start()

    mp_result_vec = []
    for _ in range(core_num):
        each_result = proc_queue.get()

        # print("mp_result_vec_object:", each_result)
        mp_result_vec.append(each_result)

    for unit_proc in proc_vec:
        unit_proc.join()
    print("exit.mp")



    #check traceplot
    grid_column= 8
    grid_row = 4
    plt.figure(figsize=(5*grid_column, 3*grid_row))
    for i, chain in enumerate(mp_result_vec):
        #plot 1
        plt.subplot(grid_row, grid_column, 4*i+1)
        plt.subplots_adjust(hspace=0.6)
        setting_contourplot(-12,12)
        chain.show_scatterplot(show=False)
        title_str = "initial: " + str(round(chain.initial[0],4)) + ", " + str(round(chain.initial[1],4)) ₩
            + "₩ngenerated sample plot"
        plt.title(title_str)

        #plot 2
        plt.subplot(grid_row, grid_column, 4*i+2)
        title_str = "total iter num:" + str(chain.num_total_iters) ₩
            + "₩nacceptance rate:" + str(round(chain.get_accept_rate(),5)) ₩
            + "₩nvisiting frequency"
        plt.title(title_str)
        chain.show_visiting_idx_hist(show=False)
```

```
        #plot 3
        plt.subplot(grid_row, grid_column, 4*i+3)
        chain.show_acf(0,10,show=False)
        title_str = "acf of x"
        plt.title(title_str)

        #plot 4
        plt.subplot(grid_row, grid_column, 4*i+4)
        chain.show_acf(1,10,show=False)
        title_str = "acf of y"
        plt.title(title_str)
    plt.show()
```

이제 코드를 돌릴 것이다. 세팅을 먼저 짚자면,

- proposal 의 bivariate normal 의 variance matrix 는 diagonal(즉 분산)에 3^2, off-diagonal(즉 공분산)은 모두 0 으로 설정하였다.
- 8 개의 chain 을 병렬로 돌릴 것이고, 각 chain 에서는 샘플 50 만개를 생성한 후, 수렴 및 acf 조정을 위해 앞 10 만개를 버리고, 200 개당 1 개의 sample 을 솎아서 최종 sample 을 얻을 것이다.
- 각 chain 의 initial point 는 random 으로 줄까 하다가 그냥 (-4,-4), (-3,-3),...(3,3) 으로 8 개를 설정하였다. random 으로 줄 시, 처음부터 파티션 범위 밖에서 시작하는 피곤한 경우가(...) 자꾸 생겼기 때문이다.

이 설정 구현은 위 multiprocessing 을 위한 함수 multiproc_1unit_do 와 main 부분에서 볼 수 있다.

아래는 실행 시 콘솔 출력이다. pid 는 os 가 할당하므로 돌릴 때마다 달라지므로 주의하라. 8 개 chain 은 동시에 만들어지므로 마지막 완료된 chain 의 실행시간이 총 실행시간이다.

```
[Console]
$ C:/Users/Rjun/AppData/Local/Programs/Python/Python37/python.exe c:/gitProject/statComputing2/HW8/HW8_SAMC.py
start.mp
pid:   10396 start!
pid:   12700 start!
pid:   6728 start!
pid:   1944 start!
pid:   2148 start!
pid:   12680 start!
pid:   12840 start!
pid:   10036 start!
(생략)
pid: 1944 iteration 500000 / 500000   done! (elapsed time for execution:   0.0 min   10.249366998672485 sec)
pid:   1944   end!
pid: 12840 iteration 500000 / 500000   done! (elapsed time for execution:   0.0 min   10.38967776298523 sec)
pid:   12840   end!
pid: 12700   iteration 450000 / 500000
pid: 6728 iteration 500000 / 500000   done! (elapsed time for execution:   0.0 min   10.67542052268982 sec)
pid:   6728   end!
pid: 12680   iteration 450000 / 500000
pid: 10396 iteration 500000 / 500000   done! (elapsed time for execution:   0.0 min   10.762267112731934 sec)
pid:   10396   end!
pid: 10036   iteration 450000 / 500000
pid: 2148 iteration 500000 / 500000   done! (elapsed time for execution:   0.0 min   10.862050294876099 sec)
pid:   2148   end!
```

```
pid: 12700 iteration 500000 / 500000    done! (elapsed time for execution:    0.0 min    11.501182794570923 sec)
pid:    12700    end!
pid: 12680 iteration 500000 / 500000    done! (elapsed time for execution:    0.0 min    11.784777402877808 sec)
pid:    12680    end!
pid: 10036 iteration 500000 / 500000    done! (elapsed time for execution:    0.0 min    11.880783081054688 sec)
pid:    10036    end!
exit.mp
```

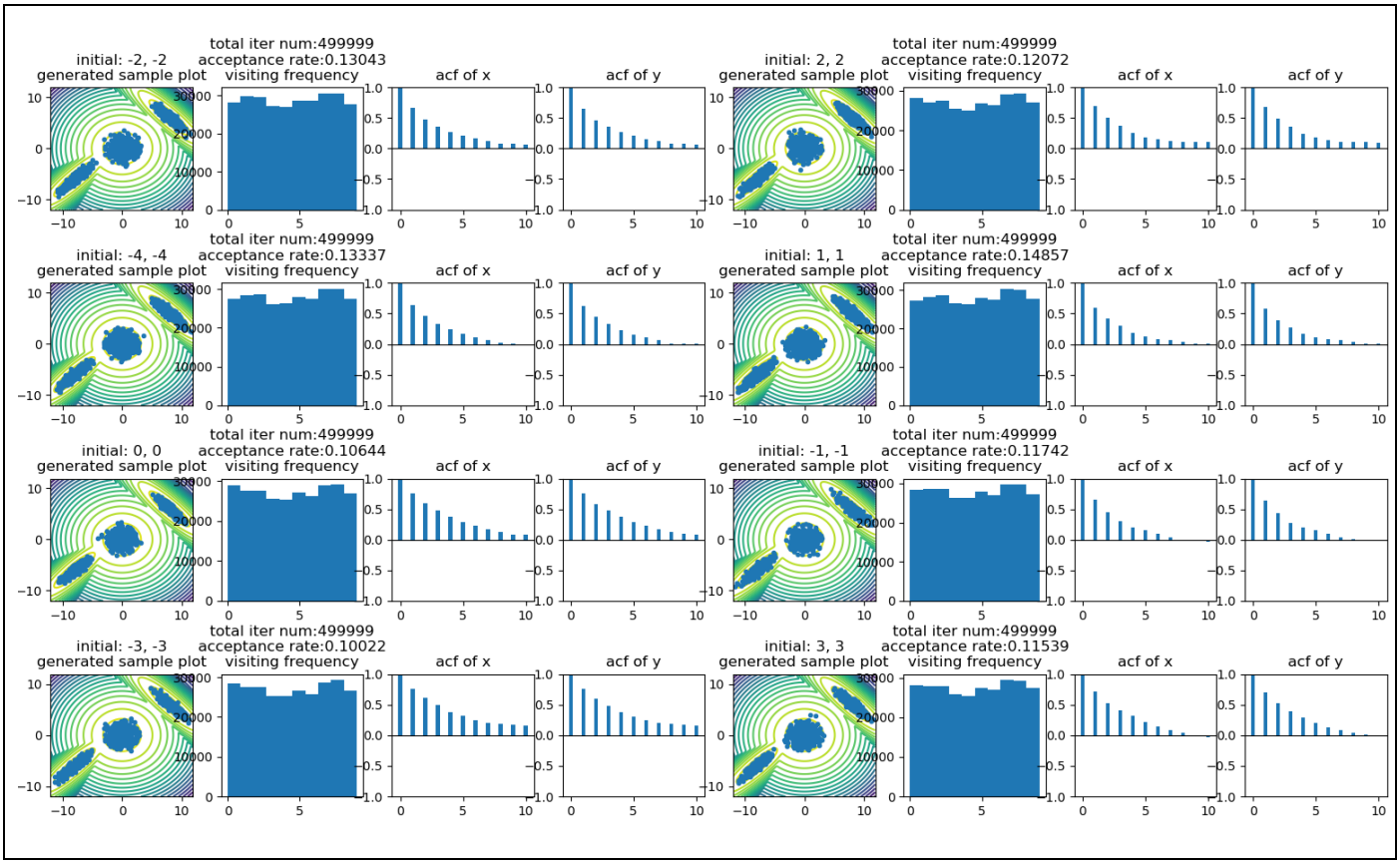
12 초만에 8 개 chain 의 각 50 만개 sample 생성이 모두 완료되었다. (빨라서 좋다!)

함께 그려지는 그래프 출력은 각 체인마다 4 개씩이고, 차례대로 다음과 같다.

1. scatterplot of generated samples on 2D contour plot of target distribution
2. momentum (over each HMC sample index)
3. autocorrelation (x 축 방향)
4. autocorrelation (Y 축 방향)

그리고 각 chain 정보를 4 개의 그래프 위에 출력하였다.

(원래는 scatterplot 대신 traceplot 을 그리려 했으나, 체인이 너무 길어 trace 를 잇는 선이 너무 난잡해지는 관계로, scatterplot 으로 대체하였다.)



그래프 출력을 보면, 모든 initial point 에서 무난히 잘 동작했음을 관찰할 수 있다. scatterplot 을 보면 모든 경우에서 3 개의 mode 를 다 찾은 것을 볼 수 있다. 또한 visiting frequency 를 보면, SAMC 알고리즘을 따라 10 개의 파티션을 나름 비슷비슷하게 proposal 했음을 알 수 있다. 덕분에 SAMC 알고리즘이 제대로 동작했다고 진단해볼 수 있다. 마지막으로, acf 도 양쪽 축으로 볼 때 잘 줄어드는 것을 볼 수 있다. 만약 thinning 을 더 과격하게 했다면, acf 를 더 앞쪽 lag 에서 죽일 수 있었을 듯하다.

SAMC 가, 같은 문제를 시도했던 숙제 7 의 HMC 결과보다 훨씬 나은 결과를 보였다.