STA414 - Probabilistic Learning

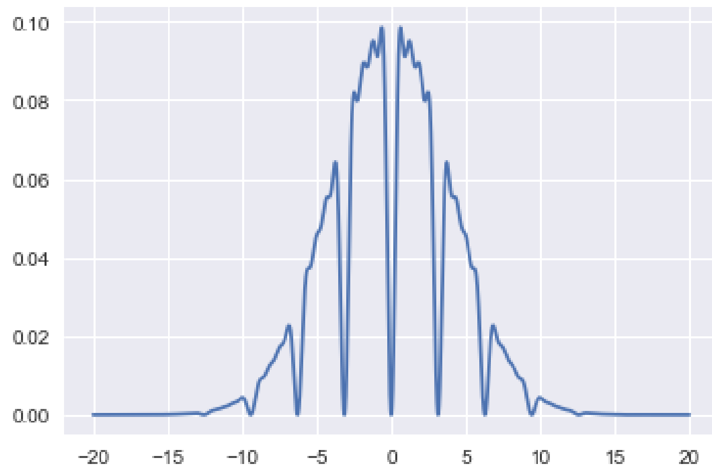
Assignment 3
Sampling and Gradient Estimators

Zikun Chen
1001117882
April 5, 2019

1 Nature's Rejection Sampler

(a) Numerical Integration

Unormalized probability $p(x, g = 0|\theta = 0)$, shown below as a function of x from -20 to 20:



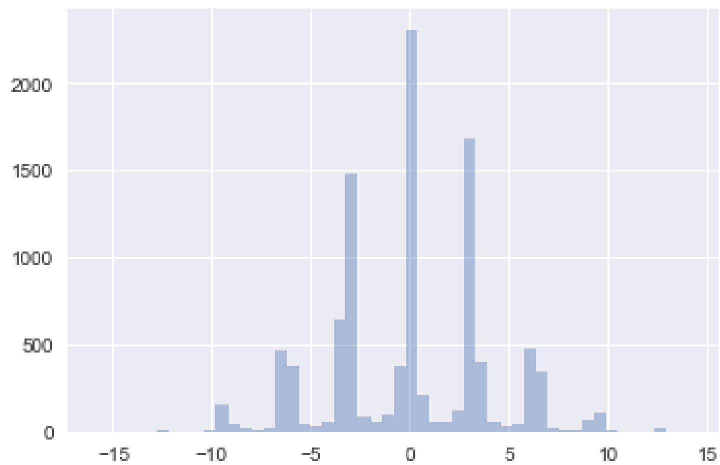
Using the quadrature function, we estimate that $p(g = 0|\theta = 0) = 0.803$

(b) Rejection Sampling

Choose proposal normal density $q(x) = p(x|\theta = 0)$ and the unormalized probability $p(x) = p(g = 1, x|\theta = 0)$.
Furthermore, it is also appropriate to use M = 1 to be the bound for $\frac{p(x)}{q(x)}$, since:

$$\frac{p(x)}{q(x)} = \frac{p(g = 1, x|\theta = 0)}{p(x|\theta = 0)} = p(g = 1|x, \theta = 0) \leq 1$$

Histogram of 10000 accepted samples:




```
Rejection Sampling:
Fraction of accepted sample is 0.803
```

(c) Importance Sampling

We want to estimate:

$$p(g=0|\theta=0) = \int p(g=0,x|\theta=0)p(x|\theta=0)dx$$

In this case, $f(x) = p(g=0,x|\theta=0)$, our target distribution is $\tilde{p}(x) = p(x|\theta = 0)$ This is the same as the proposal density $q(x) = p(x|\theta=0)$
Therefore:

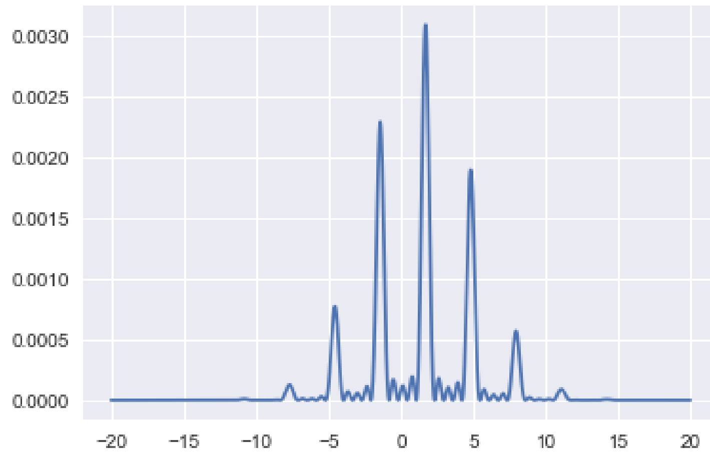
$$\hat{e}(x_1, x_2, \ldots, x_K) = \frac{1}{K}\sum_{i=1}^{K} f(x_i)\frac{\frac{\tilde{p}(x_i)}{q(x_i)}}{\frac{1}{K}\sum_{j=1}^{K}\frac{\tilde{p}(x_j)}{q(x_j)}} = \sum_{i=1}^{K} f(x_i)\frac{1}{\sum_{j=1}^{K} 1} = \frac{1}{K}\sum_{i=1}^{K} f(x_i)$$

$$= \frac{1}{K}\sum_{i=1}^{K} p(g=0,x_i|\theta=0) = \frac{1}{K}\sum_{i=1}^{K}[1 - \frac{\sin^2(5(x_i))}{25\sin^2(x_i)}] \qquad \text{where } x_i \sim_{\text{iid}} N(0,4^2)$$

This is the same as a simple monte carlo sampling of $p(g=0,x_i|\theta=0)$

```
Importance Sampling:
Estimate of fraction of photons that get absorbed is 0.802
```

(d) Plot unormalized density $p(x = 1.7, g = 1, \theta)$

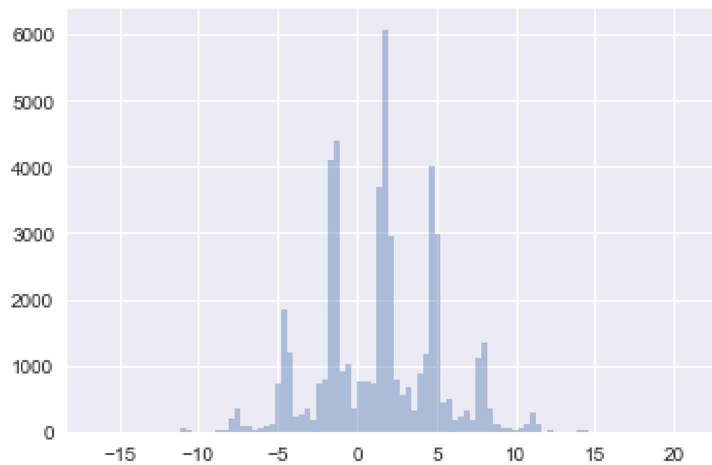


(e) Metropolis Hasting

Implementation:

```
def metro_hast(prop_sigma, N, seed, sigma=4, x=1.7, g=1):
    np.random.seed(seed)
    t = np.random.uniform(-20,20,1)[0]
    i = 0
    count = 0
    thetas = np.empty(N)
    while i < N:
        tnew = np.random.normal(t, prop_sigma, 1)[0]
        gtnew_t = st.norm.pdf(tnew, loc=t, scale=prop_sigma)
        gt_tnew = st.norm.pdf(t, loc=tnew, scale=prop_sigma)
        pt = p_theta(t, x, g)
        ptnew = p_theta(tnew, x, g)
        accept = min(1, (ptnew * gt_tnew)/(pt * gtnew_t))
        u = np.random.uniform(0,1,1)[0]
        if u < accept:
            t = tnew
            thetas[i] = t
            i += 1
        count += 1
    return thetas, acceptance
```

After experimenting with a combination of different standard deviations and sample sizes, a standard deviation of 3 is chosen for the proposal distribution and 50000 samples are drawn, which are shown below:



(f) Estimate posterior probability from samples

$$p(-3 < \theta < 3 | x = 1.7, g = 1) = \int_{-3}^{3} p(\theta | x = 1.7, g = 1) d\theta$$

We calculate the proportion of values between -3 and 3 in the samples that we drew from part e):

```
p_abs_three =  len(sample_t[(sample_t<3) & (sample_t>-3)]) \
                                         / len(sample_t)

>>> Posterior probability of theta between +3 and -3 is 0.520
```

2 Gradient Estimators

(a) Score Function has Zero Expectation

$$\begin{aligned}
\mathbb{E}_{p(x|\theta)} \left[\nabla_\theta \log p(x|\theta)\right] &= \mathbb{E}_{p(x|\theta)} \left[\frac{\nabla_\theta p(x|\theta)}{p(x|\theta)}\right] \\
&= \int p(x|\theta) \frac{\nabla_\theta p(x|\theta)}{p(x|\theta)} dx \\
&= \nabla_\theta \int p(x|\theta) dx = \nabla_\theta 1 = 0
\end{aligned}$$

(b) REINFORCE is unbiased

$$\begin{aligned}
\mathbb{E}_{p(b|\theta)}\left[f(b)\frac{\partial}{\partial\theta}\log p(b|\theta)\right] &= \mathbb{E}_{p(b|\theta)}\left[f(b)\frac{\frac{\partial}{\partial\theta}p(b|\theta)}{p(b|\theta)}\right] \\
&= \int f(b)\frac{\partial}{\partial\theta}p(b|\theta)db \\
&= \frac{\partial}{\partial\theta}\int f(b)p(b|\theta)db = \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}\left[f(b)\right]
\end{aligned}$$

(c) REINFORCE with fixed baseline is unbiased

Show the estimator is unbiased:

$$\begin{aligned}
\mathbb{E}_{p(b|\theta)}\left[c\frac{\partial}{\partial\theta}\log p(b|\theta)\right] &= c\,\mathbb{E}_{p(b|\theta)}\left[\frac{\frac{\partial}{\partial\theta}p(b|\theta)}{p(b|\theta)}\right] \\
&= c\int \frac{\partial}{\partial\theta}p(b|\theta)db \\
&= c\frac{\partial}{\partial\theta}\int p(b|\theta)db = 0
\end{aligned}$$

Therefore,

$$\mathbb{E}_{p(b|\theta)}\left[[f(b)-c]\,\frac{\partial}{\partial\theta}\log p(b|\theta)\right] = \mathbb{E}_{p(b|\theta)}\left[f(b)\frac{\partial}{\partial\theta}\log p(b|\theta)\right] - \mathbb{E}_{p(b|\theta)}\left[c\frac{\partial}{\partial\theta}\log p(b|\theta)\right] = 0$$

(d) Biased REINFORCE with dependant baseline

Show estimator is unbiased:

$$\begin{aligned}
\mathbb{E}_{p(b|\theta)}\left[[f(b)-c(b)[\,\frac{\partial}{\partial\theta}\log p(b|\theta)\right] &= \mathbb{E}_{p(b|\theta)}\left[f(b)\frac{\partial}{\partial\theta}\log p(b|\theta)\right] - \mathbb{E}_{p(b|\theta)}\left[c(b)\frac{\partial}{\partial\theta}\log p(b|\theta)\right] \\
\text{by part b)} &= \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}\left[f(b)\right] - \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}\left[c(b)\right] \\
&\neq \frac{\partial}{\partial\theta}\mathbb{E}_{p(b|\theta)}\left[f(b)\right]
\end{aligned}$$

3 Comparing variances of gradient estimators

(a) Variance of Monte Carlo Estimator

$$\mathbb{V}\left[\hat{L}_{MC}\right] = \mathbb{V}\left[\sum_{i=1}^{D} x_d\right] = \sum_{i=1}^{D}\mathbb{V}\left[x_d\right] + \sum_{i\neq j} Cov(x_i, x_j) = \sum_{i=1}^{D} 1 = D$$

(b) REINFORCE estimator with a baseline

Let $x = [x_1, \cdots, x_n]^T$ and $\theta = [\theta_1, \cdots, \theta_n]^T$

$$\log p(x|\theta) = \log \frac{1}{\sqrt{2\pi^D |I|}} exp\{-\frac{1}{2}(x-\theta)^T(x-\theta)\}$$

$$\begin{aligned}
\hat{g}^{\mathrm{SF}}[f] &= [f(x) - c(\theta)] \frac{\partial}{\partial \theta} \log p(x|\theta) \\
&= \left[\sum_{d=1}^{D} x_d - \sum_{d=1}^{D} \theta_d\right] \nabla_\theta \log \frac{1}{\sqrt{2\pi^D |I|}} exp\{-\frac{1}{2}(x-\theta)^T(x-\theta)\} \\
&= \left[\sum_{d=1}^{D} (x_d - \theta_d)\right] (-\frac{1}{2}) \nabla_\theta (x-\theta)^T(x-\theta) \\
&= \left[\sum_{d=1}^{D} (x_d - \theta_d)\right] (-\frac{1}{2})(-2)(x-\theta) \\
&= \left[\sum_{d=1}^{D} (x_d - \theta_d)\right] (x-\theta) \\
&= \epsilon(\epsilon^T \mathbb{1})
\end{aligned}$$

where $\epsilon = [\epsilon_1, \cdots, \epsilon_D]^T \sim N(0, I)$ and $\mathbb{1} = [1, \cdots, 1]^T$

(c) Variance of REINFORCE estimator

For any $d \in \{1, \cdots, D\}$:

$$E\left[\epsilon_d^2\right] = \mathbb{V}[\epsilon_d] + E\left[\epsilon_d\right]^2 = 1$$

$$
\begin{aligned}
\mathbb{V}\left[\hat{g}_1^{\text{SF}}\right] &= \mathbb{V}\left[\epsilon_1(\epsilon^T\mathbb{1})\right] \\
&= E\left[\epsilon_1^2(\epsilon^T\mathbb{1})^2\right] - E\left[\epsilon_1(\epsilon^T\mathbb{1})\right]^2 \\
&= E\left[\epsilon_1^2(\sum_{d=1}^{D}\epsilon_d)^2\right] - E\left[\epsilon_1\sum_{d=1}^{D}\epsilon_d\right]^2 \\
&= E\left[\epsilon_1^2(\sum_{d=1}^{D}\epsilon_d^2 + 2\sum_{i<j}^{D}\epsilon_i\epsilon_j)\right] - E\left[\epsilon_1^2 + \epsilon_1\sum_{d=2}^{D}\epsilon_d\right]^2 \\
&= E\left[\epsilon_1^4\right] + E\left[\epsilon_1^2\sum_{d=2}^{D}\epsilon_d^2\right] + 2\sum_{i<j}^{D}E\left[\epsilon_i\epsilon_j\right] - (E\left[\epsilon_1^2\right] + 0)^2 \\
&= 3 + E\left[\epsilon_1^2\right]E\left[\sum_{d=2}^{D}\epsilon_d^2\right] + 0 - 1^2 \\
&= 3 + (D-1) - 1 = D + 1
\end{aligned}
$$

(d) Reparameterization of gradient estimator

$$
\begin{aligned}
\nabla_\theta L(\theta) &= \nabla_\theta E_{x\sim p(x|\theta)}\left[f(x)\right] \\
&= \nabla_\theta E_{x\sim p(\epsilon)}\left[f(T(\theta,\epsilon))\right] \\
&= E_{x\sim p(\epsilon)}\left[\nabla_\theta f(T(\theta,\epsilon))\right] \\
&= E_{x\sim p(\epsilon)}\left[\nabla_\theta f(\theta+\epsilon)\right] \\
&= E_{x\sim p(\epsilon)}\left[\nabla_\theta\sum_{d=1}^{D}[\theta_d+\epsilon_d]\right] \\
&= E_{x\sim p(\epsilon)}\left[\nabla_\theta\mathbb{1}\right] \\
&= E_{x\sim p(\epsilon)}\left[\mathbb{1}\right] = \mathbb{1}
\end{aligned}
$$

$$
\hat{g}^{\text{REPARAM}} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial\theta} = \mathbb{1}^T I = \mathbb{1}^T
$$

$$
\mathbb{V}\left[\hat{g}_1^{\text{REPARAM}}\right] = \mathbb{V}\left[1\right] = 0 < \mathbb{V}\left[\hat{g}_1^{\text{SF}}\right]
$$

Again $\mathbb{1} = [1,\cdots,1]^T$

4 Appendix: Code

```
import numpy as np
import seaborn as sns
import scipy.stats as st
import scipy
import matplotlib.pyplot as plt
```

```python
sns.set()

def pg(x, g, theta=0, sigma=4):
    result = scipy.sin(5*(x-theta))**2/(25*scipy.sin(x-theta)**2)
    return result if g == 1 else (1-result)

def px(x, theta=0, sigma=4):
    return st.norm.pdf(x, loc=theta, scale=sigma)

def smc(g, N, seed, theta=0, sigma=4):
    np.random.seed(SEED)
    X = np.random.normal(theta, sigma, N)
    return np.sum([pg(x, g) for x in X])/N

def rejection_sampler(M, N, seed, theta=0, sigma=4):
    np.random.seed(SEED)
    i = 0
    count = 0
    samples = np.empty(N)
    while i < N:
        x = np.random.normal(theta, sigma, 1)[0]
        u = np.random.uniform(0,1,1)[0]
        q_x = px(x, theta, sigma)
        p_x = pg(x, 1, theta, sigma)*q_x
        if u < p_x/(M*q_x):
            samples[i] = x
            i += 1
        count += 1
    acceptance = N / count
    return samples, acceptance

def importance_sampler(N, seed):
    return smc(0, N, seed)


def p_theta(theta, x=1.7, g=0, sigma=4):
    z = 1 + (theta/10)**2
    result = px(x, theta, sigma) * pg(x, g, theta, sigma)
    return result * 1 / (10 * np.pi * z)


def metro_hast(prop_sigma, N, seed, sigma=4, x=1.7, g=1):
    np.random.seed(seed)
    t = np.random.uniform(-20,20,1)[0]
    i = 0
```

```
    count = 0
    thetas = np.empty(N)
    while i < N:
        tnew = np.random.normal(t, prop_sigma, 1)[0]
        gtnew_t = st.norm.pdf(tnew, loc=t, scale=prop_sigma)
        gt_tnew = st.norm.pdf(t, loc=tnew, scale=prop_sigma)
        pt = p_theta(t, x, g)
        ptnew = p_theta(tnew, x, g)
        accept = min(1, (ptnew * gt_tnew)/(pt * gtnew_t))
        u = np.random.uniform(0,1,1)[0]
        if u < accept:
            t = tnew
            thetas[i] = t
            i += 1
        count += 1
    return thetas, acceptance


if __name__ == "__main__":
    theta = 0
    sigma = 4
    N = 10000
    SEED = 2019

# 1 a)
    # Quadrature (Direct Integration)
    x = np.linspace(start=-20, stop=20, num=N)
    f = lambda x: px(x) * pg(x, 0)

    p_absorb = scipy.integrate.quadrature(f, -20, 20)[0]
    plt.plot(x, f(x))
    plt.show()

    print('Quadrature estimate of fraction of photons absorbed', \
          'average is %.3f' % (p_absorb))

# 1 b)
    samples, acceptance = rejection_sampler(1, 10000, SEED)

    sns.distplot(samples, kde = False)
    plt.show()
    print('Rejection Sampling: ')
    print('Fraction of accepted sample is %.3f' % (acceptance))


# 1 c)
```

```python
    p_absorb_marginal = importance_sampler(1000, SEED)
    print('Importance Sampling:')
    print('Estimate of fraction of photons that get absorbed is %.3f'\
        % (p_absorb_marginal))

# 1 d)
    x = 1.7
    g = 1
    thetas = np.linspace(start=-20, stop=20, num=N)
    p_joint = p_theta(thetas, x, g)
    plt.plot(thetas, p_joint)
    plt.show()

# 1 e)
    sample_t, acceptance = metro_hast(3, 50000, SEED)
    sns.distplot(sample_t, kde = False)
    plt.show()

    print('Metropolis Hasting to sample theta:')
    print('Fraction of accepted sample is %.3f' % (acceptance))

# 1 f)
    p_abs_three =  len(sample_t[(sample_t<3) & (sample_t>-3)]) \
                                / len(sample_t)

    print('Posterior probability of theta between' \
          ' +3 and -3 is %.3f' % (p_abs_three))
```