# Homework 1

Jingwei Zhang 201528013229095

2015-10-15

## 1 Problem 1

题目1:

(a): $\hat{\mu}_n$复杂度为$\theta(dn)$，$C_n$复杂度为$\theta(d^2n)$

(b): $\hat{\mu}_{n+1} = \frac{1}{n+1}\sum_{k=1}^{n+1} x_k = \frac{n}{n+1}\cdot\frac{1}{n}\sum_{k=1}^{n} x_k + \frac{1}{n+1}x_{k+1} = (1-\frac{1}{n+1})\hat{\mu}_n + \frac{1}{n+1}x_{k+1} = \hat{\mu}_n + \frac{1}{n+1}(x_{n+1}-\hat{\mu}_n)$

$$C_{n+1} = \frac{1}{n}\sum_{i=1}^{n+1}(x_k-\hat{\mu}_{n+1})(x_k-\hat{\mu}_{n+1})^T = \frac{1}{n}\sum_{i=1}^{n+1}\left[x_k-\hat{\mu}_n-\frac{1}{n+1}(x_{k+1}-\hat{\mu}_n)\right]\left[x_k-\hat{\mu}_n-\frac{1}{n+1}(x_{k+1}-\hat{\mu}_n)\right]^T$$

$$= \frac{n-1}{n}\cdot\frac{1}{n-1}\sum_{i=1}^{n}(x_k-\hat{\mu}_n)(x_k-\hat{\mu}_n)^T - \frac{1}{n}\sum_{i=1}^{n}(x_k-\hat{\mu}_n)\cdot\frac{1}{n+1}(x_{k+1}-\hat{\mu}_n)^T - \frac{1}{n}\cdot\frac{1}{n+1}(x_{n+1}-\hat{\mu}_n)\sum_{k=1}^{n}(x_k-\hat{\mu}_n)^T$$

$$+\frac{1}{n}\left[(x_{n+1}-\hat{\mu}_n)-\frac{1}{n+1}(x_{n+1}-\hat{\mu}_n)\right]\cdot\left[(x_{n+1}-\hat{\mu}_n)-\frac{1}{n+1}(x_{n+1}-\hat{\mu}_n)\right]^T + \frac{1}{(n+1)^2}(x_{n+1}-\hat{\mu}_n)(x_{n+1}-\hat{\mu}_n)^T$$

$$= \frac{n-1}{n}\cdot C_n - 0 - 0 + \frac{1}{(n+1)^2}(x_{n+1}-\hat{\mu}_n)(x_{n+1}-\hat{\mu}_n)^T + \frac{n}{(n+1)^2}(x_{n+1}-\hat{\mu}_n)(x_{n+1}-\hat{\mu}_n)^T$$

$$= \frac{n-1}{n}C_n + \frac{1}{n+1}(x_{n+1}-\hat{\mu}_n)(x_{n+1}-\hat{\mu}_n)^T$$

(c) 每一步(迭代)计算复杂度为 $\hat{\mu}_n:\theta(d)$，$C_n:\theta(d^2)$

计算到n(总共)计算复杂度为 $\hat{\mu}_n:\theta(dn)$，$C_n:\theta(d^2n)$

(d) 当要知道n不断增长过程中每一步的μ与C时，用迭代比较好，比如，数据是动态增长的，我们需要观测μ与C及他们的变化时，用迭代比较快。

如果n是不变的，即我们只要知道$\hat{\mu}_n$与$C_n$就可以时，用非迭代比较好。

# 2 Problem 2

## 2.1 a

According to the question , we have the data set $\mathcal{D} = \left\{ \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 3 \end{pmatrix}, \begin{pmatrix} 2 \\ \star \end{pmatrix} \right\}$, we have :

$$\begin{aligned} Q(\theta, \theta^0) =& E_{x_{32}} \left[ \ln P(x_g, x_b; \theta) | \theta^0, \mathcal{D} \right] \\ =& \int_{-\infty}^{+\infty} [\ln P(x_1|\theta) + \ln P(x_2|\theta) + \ln P(x_3|\theta)] P(x_{32}|\theta^0, x_{31} = 2) dx_{32} \\ =& \ln P(x_1|\theta) + \ln P(x_2|\theta) + \int_{-\infty}^{+\infty} \ln P(x_3|\theta) P(x_{32}|\theta^0, x_{31} = 2) dx_{32} \\ =& \ln P(x_1|\theta) + \ln P(x_2|\theta) + 2e \int_{-\infty}^{+\infty} \ln P \left[ \begin{pmatrix} 2 \\ x_{32} \end{pmatrix} \right] P \left[ \begin{pmatrix} 2 \\ x_{32} \end{pmatrix} |\theta^0 \right] dx_{32} \end{aligned}$$

Then, let $T = 2e \int_{-\infty}^{+\infty} \ln P \left[ \begin{pmatrix} 2 \\ x_{32} \end{pmatrix} \right] P \left[ \begin{pmatrix} 2 \\ x_{32} \end{pmatrix} |\theta^0 \right] dx_{32}$, we have:

**1.** If $3 \leq \theta_2 \leq 4$ :

$$\begin{aligned} T =& \frac{1}{4} \int_0^{\theta_2} \ln \left( \frac{1}{\theta_1} e^{-2\theta_1} \times \frac{1}{\theta_2} \right) dx_{32} \\ =& \frac{1}{4} \theta_2 \ln \left( \frac{1}{\theta_1} e^{-2\theta_1} \times \frac{1}{\theta_2} \right) \end{aligned}$$

**2.** If $4 < \theta_2$ :

$$\begin{aligned} T =& \frac{1}{4} \int_0^{4} \ln \left( \frac{1}{\theta_1} e^{-2\theta_1} \times \frac{1}{\theta_2} \right) dx_{32} \\ =& \ln \left( \frac{1}{\theta_1} e^{-2\theta_1} \times \frac{1}{\theta_2} \right) \end{aligned}$$

**3.** If $\theta < 3$ :

$$T = 0$$

Therefore,we get :

$$\begin{aligned} Q(\theta, \theta^0) =& \ln P(x_1|\theta) + \ln P(x_2|\theta) + T \\ =& -4\theta_1 - 2\ln(\theta_1\theta_2) + T \end{aligned}$$

Because of :

$$\int_{-\infty}^{+\infty} P(x_1) dx_1 = 1$$

$$\int_{-\infty}^{+\infty} \frac{1}{\theta_1} e^{-\theta_1 x_1} dx_1 = 1$$

Thus, above all we get the result :

$$\theta_1 = 1$$

### 2.2 b

**(1)** $3 \le \theta_2 \le 4$,according to the formula of $Q(\theta,\theta^0)$,we get:

$$Q(\theta,\theta^0) = -4 - \left(2\ln\theta_2 + \frac{1}{4}\theta_2(2+\ln\theta_2)\right)$$

Therefore, when $\theta_2 = 3$ ,we get the max result,$Q(\theta,\theta^0) = -8.521$

**(2)** $\theta_2 > 4$,according to the formula of $Q(\theta,\theta^0)$, we get:

$$Q(\theta,\theta^0) = -6 - 3\ln\theta_2$$

Therefore, when $\theta_2 = 4$ ,we get the max result,$Q(\theta,\theta^0) = -10.159$ Thus , above all ,the result is $\theta = (3 \quad 4)^t$.

## 3 Problem 3

### 3.1 a

$$\begin{aligned}
\bar{p_n}(x) &= E[p_n(x)] \\
&= \frac{1}{nh_n}\sum_{i=1}^{n} E\left[\varphi\left(\frac{x-x_i}{h_n}\right)\right] \\
&= \frac{1}{h_n}\int \varphi\left(\frac{x-v}{h_n}\right)p(v)dv \\
&= \frac{1}{h_n}\int \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-v)^2}{2h_n^2} - \frac{(v-\mu)^2}{2\sigma^2}\right]dv \\
&= \frac{1}{2\pi\sigma h_n}\exp\left[-\frac{1}{2}\left(\frac{x^2}{h_n^2}+\frac{\mu^2}{\sigma^2}\right)\right]\int \exp\left[-\frac{1}{2}\left(\frac{1}{h_n^2}+\frac{1}{\sigma^2}\right)v^2 + \left(\frac{x}{h_n^2}+\frac{\mu}{\sigma^2}\right)v\right]dv \\
&= \frac{\sigma'}{\sqrt{2\pi}h_n\sigma}\exp\left[-\frac{1}{2}\left(\frac{x^2}{h_n^2}+\frac{\mu^2}{\sigma^2}\right)+\frac{1}{2}\frac{\mu'^2}{\sigma'^2}\right]\int \frac{1}{\sqrt{2\pi}\sigma'}\exp\left[-\frac{1}{2}\left(\frac{v-\mu'}{\sigma'}\right)^2\right]dv \\
&= \frac{\sigma'}{\sqrt{2\pi}h_n\sigma}\exp\left[-\frac{1}{2}\left(\frac{x^2}{h^2}+\frac{\mu^2}{\sigma^2}-\frac{\mu'^2}{\sigma'^2}\right)\right] \\
&= \frac{1}{\sqrt{2\pi}h_n\sigma}\frac{h_n\sigma}{\sqrt{h_n^2+\sigma^2}}\exp\left[-\frac{1}{2}\left(\frac{x^2\sigma^2h_n^2+x^2\sigma^4+h_n^4\mu^2+h_n^2\mu^2\sigma^2}{h_n^2\sigma^2(h_n^2+\sigma^2)} - \frac{x^2\sigma^4+h_n^4\mu^2+2x\sigma^2h_n^2\mu}{h_n^2\sigma^2(h_n^2+\sigma^2)}\right)\right] \\
&= \frac{1}{\sqrt{2\pi}}\frac{1}{\sqrt{h_n^2+\sigma^2}}\exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{h_n^2+\sigma^2}\right]
\end{aligned}$$

$$\begin{cases} \sigma'^2 = \frac{h_n^2\sigma^2}{h_n^2+\sigma^2} \\ \mu' = \left(\frac{x}{h_n^2}+\frac{\mu}{\sigma^2}\right)\sigma'^2 \end{cases}$$

Thus, $\bar{p_n}(x) \sim N(\mu, \sigma^2 + h_n^2)$

# 4 Problem 5

## 4.1 Result

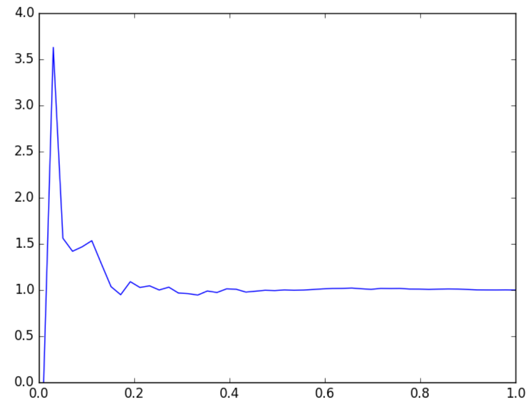


Figure 1: Figure for Problem b (uniform distribution)

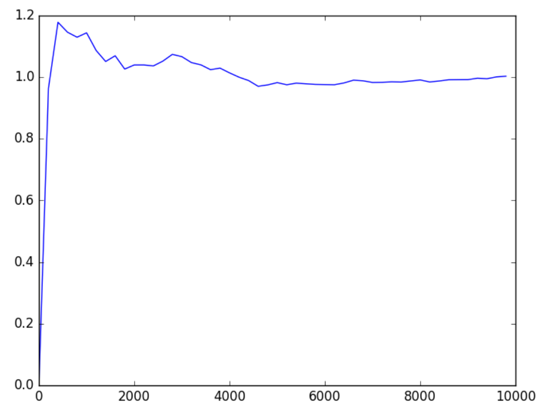


Figure 2: Figure for Problem c (uniform distribution)

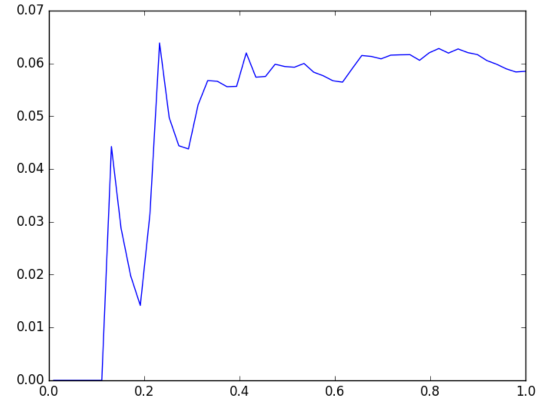


Figure 3: Figure for Problem b (gaussian distribution)

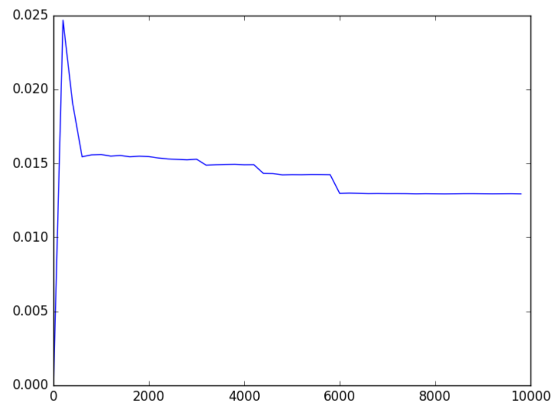


Figure 4: Figure for Problem c (gaussian distribution)

For uniform distribution two method all goes to $p((0,0,0)) = 1$ when n increase to sufficiently large and its convergence is quick. However, For gaussian distribution, its convergence is much slower than that of uniform distribution.

## 4.2 Code

```
#!/usr/bin/python
# coding=utf-8

import numpy as np
import matplotlib.pyplot as plt

# constant values
N = 10000
X_L = -1.0/2
X_R = 1.0/2
orgin = [0, 0, 0]

def phi(x,xi,h):
    for i in range(len(x)):
```

```python
        if np.abs((x[i] - xi[i]) / h) > 1.0 / 2:
            return 0
    else:
        return 1

# Problem a)
def gen_uniform_dis():
    l = np.random.uniform(low = X_L,high = X_R, size=(N,3))
    return l

# Problem b)
def parzen_window_estimation(points, w_size):
    h = w_size
    n = len(points)
    kn = np.sum([phi(orgin, x, h) for x in points])
    p = kn / n / (h**len(points[0]))
    return p

def parzen_window(points):
    x = np.linspace(0.01, 1)
    y = [parzen_window_estimation(points, w) for w in x]
    return x,y

# Problem c)
def window_estimation(points, n):
    h = np.max([np.max([np.abs(x) for x in points[i]]) for i in range(n)])
    # h = h * 2
    kn = np.sum([phi(orgin, points[i], h) for i in range(n)])
    p = kn / n / (h**len(points[0]))
    return p

def window(points, step):
    x = [i for i in range(1,N,step)]
    y = [window_estimation(points,i) for i in x]
    return x,y

# Problem d)
def gen_gaussian_dis():
    l = np.random.normal(0,1, size=(N,3))
    return l

if __name__ == '__main__':
    # Problem a
    ud_points = gen_uniform_dis()
    # Problem b
    x,y = parzen_window(ud_points)
    line, = plt.plot(x,y)
    plt.show()
    # Problem c
    x, y = window(ud_points, 200)
    line, = plt.plot(x,y)
    plt.show()
    # Problem d
    norm_points = gen_gaussian_dis()
    x,y = parzen_window(norm_points)
    line, = plt.plot(x,y)
```

```
plt.show()
x, y = window(norm_points, 200)
line, = plt.plot(x,y)
plt.show()
```

# 5 Problem 5

## 5.1 Result

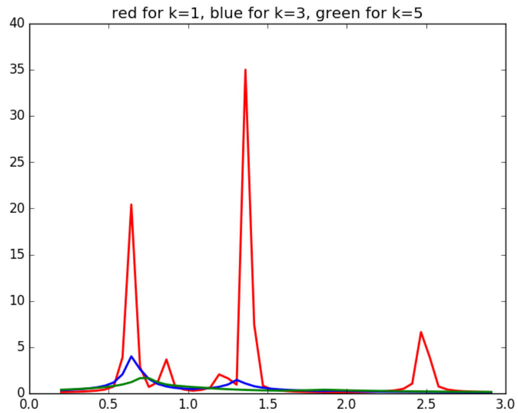


Figure 5: Figure for Problem a

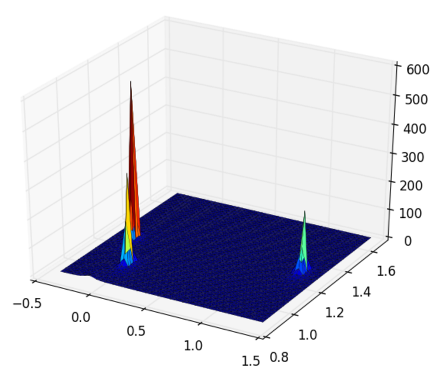

Figure 6: Figure for Problem b, k = 1

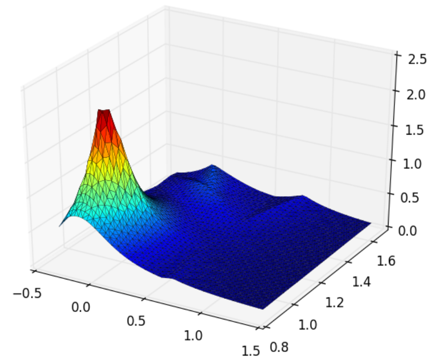


Figure 7: Figure for Problem b, k = 3

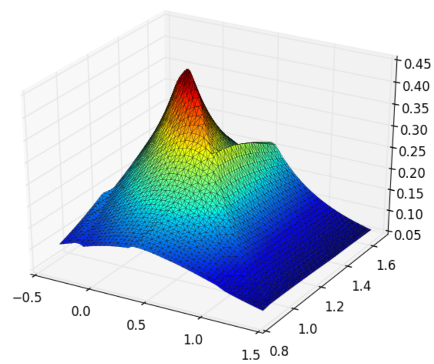


Figure 8: Figure for Problem b, k = 5

## 5.2 Code

```
#!/usr/bin/python
# coding=utf-8
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np
import matplotlib.pyplot as plt
import math

# constant values
gap = 0.4
EPS = 1e-9
all_k = [1, 3, 5]
c1 = np.array([
    [0.28, 1.31, -6.2],
    [0.07, 0.58, -0.78],
    [1.54, 2.01, -1.63],
    [-0.44, 1.18, -4.32],
```

```
    [-0.81, 0.21, 5.73],
    [1.52, 3.16, 2.77],
    [2.20, 2.42, -0.19],
    [0.91, 1.94, 6.21],
    [0.65, 1.93, 4.38],
    [-0.26, 0.82, -0.96]
])
c2 = np.array([
    [0.011, 1.03, -0.21],
    [1.27, 1.28, 0.08],
    [0.13, 3.12, 0.16],
    [-0.21, 1.23, -0.11],
    [-2.18, 1.39, -0.199],
    [0.34, 1.96, -0.16],
    [-1.38, 0.94, 0.45],
    [-0.12, 0.82, 0.17],
    [-1.44, 2.31, 0.14],
    [0.26, 1.94, 0.08]
])
c3 = np.array([
    [1.36, 2.17, 0.14],
    [1.41, 1.45, -0.38],
    [1.22, 0.99, 0.69],
    [2.46, 2.19, 1.31],
    [0.68, 0.79, 0.87],
    [2.51, 3.22, 1.35],
    [0.60, 2.44, 0.92],
    [0.64, 0.13, 0.97],
    [0.85, 0.58, 0.99],
    [0.66, 0.51, 0.88]
])


def dis(a, b):
    return np.sqrt(np.sum([(a[i] - b[i])**2 for i in range(len(a))]))

def getV(r, d):
    v = 0
    if d == 1:
        v = 2 * r
    elif d == 2:
        v = math.pi * r * r
    elif d == 3:
        v = 4.0 / 3.0 * math.pi * (r**3)
    return v

def knn_estimation(points, x, k):
    n = len(points)
    diss = [dis(x, p) for p in points]
    h = max(sorted(diss)[k - 1],EPS)
    v = getV(h, len(points[0]))
    p = k/n/v
    return p

# Problem a)
def problem_a():
```

```
    points = [[x[0]] for x in c3]
    x_min = np.min(points) - gap
    x_max = np.max(points) + gap
    x = np.linspace(x_min, x_max)
    ys = []
    for k in all_k:
        ys.append([knn_estimation(points, [xi], k) for xi in x])
    return x, ys

def problem_b():
    points = c2[:,(0,1)]
    d = 2
    points_x = []
    x_min = []
    x_max = []
    for i in range(d):
        points_x.append(points[:,0])
        x_min.append(np.min(points[i]) - gap)
        x_max.append(np.max(points[i]) + gap)

    single_x = []
    for i in range(d):
        single_x.append(np.linspace(x_min[i],x_max[i]))
    x = []
    for xi in single_x[0]:
        for yi in single_x[1]:
            x.append([xi,yi])

    ys = []
    for k in all_k:
        ys.append([knn_estimation(points, xi, k) for xi in x])
    return x, ys

lw = 2
colors = ['r','b','g']

if __name__ == '__main__':
    # Problem a
    x, ys = problem_a()
    plt.title('red for k=1, blue for k=3, green for k=5')
    for i in range(len(ys)):
        plt.plot(x,ys[i],color=colors[i],
            linewidth=lw, label=("k="+str(all_k[i])))
    plt.show()

    # Problem b
    x, ys = problem_b()
    x_1 = [xi[0] for xi in x]
    x_2 = [xi[1] for xi in x]
    for i in range(3):
        fig = plt.figure()
        ax = fig.gca(projection='3d')
        ax.plot_trisurf(x_1, x_2, ys[i], cmap=cm.jet, linewidth=0.2)
        plt.show()
```