STAT 580 Homework 3

Yifan Zhu

February 28, 2017

1. (a) The joint density of (X, U) is

$$f(x, u) = f(x)f(u) = g(x)$$

Then

$$\begin{aligned}
P(U \le r(X)) &= P\left(U \le \frac{q(X)}{\alpha g(X)}\right) \\
&= \int_{\mathcal{X}} \int_0^{\frac{q(x)}{\alpha g(x)}} f(x, u) \mathrm{d}u \mathrm{d}x \\
&= \int_{\mathcal{X}} \int_0^{\frac{q(x)}{\alpha g(x)}} g(x) \mathrm{d}u \mathrm{d}x \\
&= \int_{\mathcal{X}} g(x) \frac{q(x)}{\alpha g(x)} \mathrm{d}x \\
&= \frac{1}{\alpha} \int_{\mathcal{X}} q(x) \mathrm{d}x
\end{aligned}$$

(b) In the same way, we have

$$\begin{aligned}
P(X \in A, U \le r(X)) &= P\left(X \in A, U \le \frac{q(X)}{\alpha g(X)}\right) \\
&= \int_A \int_0^{\frac{q(x)}{\alpha g(x)}} f(x, u) \mathrm{d}u \mathrm{d}x \\
&= \int_A \int_0^{\frac{q(x)}{\alpha g(x)}} g(x) \mathrm{d}u \mathrm{d}x \\
&= \int_A g(x) \frac{q(x)}{\alpha g(x)} \mathrm{d}x \\
&= \frac{1}{\alpha} \int_A q(x) \mathrm{d}x
\end{aligned}$$

We know that

$$P(Y \in A) = P(X \in A | U \le r(X))$$

Then

$$P(Y \in A) = \frac{P(X \in A, U \le r(X))}{P(U \le r(X))} = \frac{\frac{1}{\alpha} \int_A q(x) \mathrm{d}x}{\frac{1}{\alpha} \int_{\mathcal{X}} q(x) \mathrm{d}x} = \int_A \left(q(x) \Big/ \int_{\mathcal{X}} q(x) \mathrm{d}x \right) \mathrm{d}x = \int_A f(x) \mathrm{d}x$$

2. (a)

$$\begin{aligned}\frac{1}{C} &= \int_0^{\infty}(2x^{\theta-1}+x^{\theta-1/2})\mathrm{e}^{-x}\mathrm{d}x \\ &= 2\int_0^{\infty} x^{\theta-1}\mathrm{e}^{-x}\mathrm{d}x + \int_0^{\infty} x^{\theta-1/2}\mathrm{e}^{-x}\mathrm{d}x \\ &= 2\Gamma(\theta)+\Gamma(\theta+1/2)\end{aligned}$$

Thus the normalizing constant

$$C = \frac{1}{2\Gamma(\theta)+\Gamma(\theta+1/2)}$$

(b)

$$\begin{aligned}g(x) &= \frac{1}{2\Gamma(\theta)+\Gamma(\theta+1/2)}\left(2x^{\theta-1}\mathrm{e}^{-x}+x^{\theta-1/2}\mathrm{e}^{-x}\right) \\ &= \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}\frac{1}{\Gamma(\theta)}x^{\theta-1}\mathrm{e}^{-x} + \frac{\Gamma(\theta+1/2)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}\frac{1}{\Gamma(\theta+1/2)}x^{\theta-1/2}\mathrm{e}^{-x} \\ &= w_1 g_1(x) + w_2 g_2(x)\end{aligned}$$

$w_1 + w_2 = 1$ amd g_1 is from $\Gamma(\theta)$ and g_2 is from $\Gamma(\theta+1/2)$.

(c) For $X_1 \sim \Gamma(\theta), X_2 \sim \Gamma(\theta+1/2), U \sim \mathrm{Unif}(0,1)$, Let

$$Z = \begin{cases} X_1, U \leq \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)} \\ X_2, U > \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)} \end{cases}$$

Then

$$\begin{aligned}P(Z\in A) &= P\left(Z\in A|U\leq \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}\right)P\left(U\leq \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}\right) \\ &\quad + P\left(Z\in A|U> \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}\right)P\left(U> \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}\right) \\ &= \frac{2\Gamma(\theta)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}P(X_1\in A) + \frac{\Gamma(\theta+1/2)}{2\Gamma(\theta)+\Gamma(\theta+1/2)}P(X_2\in A) \\ &= w_1\int_A g_1(x)\mathrm{d}x + w_2\int_A g_2(x)\mathrm{d}x \\ &= \int_A g(x)\mathrm{d}x\end{aligned}$$

Hence, Z has the desired mixture gamma distribution. Then we can generate it like this

Algorithm 1 Procedure to sample from $g(x)$

```
generate U ~ Unif(0,1);
if U ≤ 2Γ(θ)/(2Γ(θ)+Γ(θ+1/2)) then
    generate X ~ Γ(θ);
else
    generate X ~ Γ(θ+1/2);
end if
```

(d) $\frac{q(x)}{\alpha g(x)} \leq 1 \Rightarrow \alpha \geq \frac{q(x)}{g(x)}$.

$$\frac{q(x)}{g(x)} = \frac{C\sqrt{x+4}}{\sqrt{x}+2}$$

Thus $\alpha = \sup\{\frac{C\sqrt{x+4}}{\sqrt{x}+2} : x > 0\} = C$. Then

$$r(x) = \frac{q(x)}{\alpha g(x)} = \frac{\sqrt{x+4}}{\sqrt{x}+2}$$

The rejection procedure is

Algorithm 2 Rejection sampling using $g(x)$ as proposal distribution

1: generate $U \sim \mathrm{Unif}(0,1)$ and $X \sim g(x)$ independently;
2: **if** $U > r(X) = \frac{\sqrt{X+4}}{\sqrt{X}+2}$ **then**
   return to Step 1;
3: **else**
4:     return X;
5: **end if**

3.

```
#include <stdio.h>

#define N 16 /* number of observations */
#define P 2 /* number of predictors */



void dgels_(char *TRANS, int *m, int *n, int *NRHS, double *A, int *LDA,
     double *B, int *LDB, double *WORK, int *LWORK, int *INFO);

int main()
{
    /* longley dataset from R: Employed (Y) GNP.deflator and Population
        (X) */
    double Y[N] = {60.323, 61.122, 60.171, 61.187, 63.221, 63.639,
        64.989,
                   63.761, 66.019, 67.857, 68.169, 66.513, 68.655,
                       69.564,
                   69.331, 70.551
                  };
    double X[N][P] =
    {
        {83, 107.608},
        {88.5, 108.632},
        {88.2, 109.773},
        {89.5, 110.929},
        {96.2, 112.075},
        {98.1, 113.27},
        {99, 115.094},
        {100, 116.219},
        {101.2, 117.388},
        {104.6, 118.734},
        {108.4, 120.445},
        {110.8, 121.95},
        {112.6, 123.366},
        {114.2, 125.368},
        {115.7, 127.852},
        {116.9, 130.081}
    };


    char trans = 'N';
    int m = N;
    int n = P+1;
    int nrhs = 1;
    int lwork = 2 * m*n;
    double work[lwork];
    int info;
    int i, j;

    double A[m * n];
    double B[m];

    for (i=0; i<m; i++){
   A[i]=1;
```

```
}

    for (i = 0; i < m; i++)
    {
        for (j = 1; j < n+1; j++)
        {
            A[j * m + i] = X[i][j-1];
        }
    }

    for (i = 0; i < m; i++)
    {
        B[i] = Y[i];
    }

    dgels_(&trans, &m, &n, &nrhs, A, &m, B, &m, work, &lwork, &info);

    if (info != 0)
    {
        printf("dgels error %d\n", info);
    }
    else
    {
        printf("The regression coefficients: ");
        for (i = 0; i < n; i++)
        {
            printf("%.6f\t", B[i]);
        }
        printf("\n");
    }
return 0;

}
```

4.

```c
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

#define N 16 /* number of observations */
#define P 2 /* number of predictors */

void dgesvd_(char *JOBU, char *JOBVT, int *m, int *n, double *A, int *
    LDA, double *S, double *U, int *LDU, double *VT, int *LDVT, double *
    WORK, int *LWORK, int *INFO);

int main()
{
    char jobu = 'S';
    char jobvt = 'A';
    int m = N;
    int n = P;
    double A[m * n];
    double s[n];
    double u[m * n];
    double vt[n * n];
    double *work;
    int lwork = -1;
    double lworkopt;
    int i, j, info;

    /* longley dataset from R */
    double X[N][P] =
    {
        {83, 107.608},
        {88.5, 108.632},
        {88.2, 109.773},
        {89.5, 110.929},
        {96.2, 112.075},
        {98.1, 113.27},
        {99, 115.094},
        {100, 116.219},
        {101.2, 117.388},
        {104.6, 118.734},
        {108.4, 120.445},
        {110.8, 121.95},
        {112.6, 123.366},
        {114.2, 125.368},
        {115.7, 127.852},
        {116.9, 130.081}
    };

    double Xbar[P];
    for (j = 0; j < P; j++)
    {
        Xbar[j] = 0;
        for (i = 0; i < N; i++)
        {
            Xbar[j] = Xbar[j] + X[i][j];
        }
```

```
        Xbar[j] = Xbar[j] / (double) N ;
    }

    for (i = 0; i < m; i++)
    {
        for (j = 0; j < n; j++)
        {
            A[j * m + i] = X[i][j] - Xbar[j];
        }
    }

    dgesvd_(&jobu, &jobvt, &m, &n, A, &m, s, u, &m, vt, &n, &lworkopt, &
        lwork, &info);

    if (info != 0)
    {
        printf("The dgesvd error %d\n", info);
    }
    else
    {
        lwork = (int) lworkopt;
        work = (double *) malloc(lwork * sizeof(double));
        assert(work != NULL);

        dgesvd_(&jobu, &jobvt, &m, &n, A, &m, s, u, &m, vt, &n, work, &
            lwork, &info);

        if (info != 0)
        {
            printf("The dgesvd error %d\n", info);
        }
        else
        {
            printf("The principal component scores:\n");
            for (i = 0; i < m; i++)
            {
                for (j = 0; j < n; j++)
                {
                    printf("%.6f\t", u[j * m + i] * s[j]);
                }
                printf("\n");
            }
        }
    }
    return 0;
}
```

5.

```c
#include <stdio.h>

#define N 10

int main(){
    double x[N] = {3.1, -1.2, 5.3, 1, 4.4, 21, 3, 7, -1.2, 3.2};
    int i, j;
    double temp;
    for (i = 1; i < N; i++){
        j = i;
        while (j > 0 && x[j-1] > x[j]){
            temp = x[j-1];
            x[j-1] = x[j];
            x[j] = temp;
            j--;
        }
    }
    printf("Sorted␣data:\n");
    for (i=0; i < N; i++){
        printf("%f␣", x[i]);
    }

    printf("\n␣Median:\n");
    if (N%2 == 0){
        printf("%f\n", (x[N/2 -1] + x[N/2])/2.0);
    } else {
        printf("%f\n", x[N/2]);
    }
    return 0;
}
```