Stat580 - Homework 2

Alex Shum

March 5, 2014

1 Problem 1

$$f(x,y) = f(y|x)f(x) = \frac{f(y|x)}{\frac{1}{f(x)}} = \frac{f(y|x)}{\int \frac{f(y)}{f(x)}dy} = \frac{f(y|x)}{\int \frac{\frac{1}{f(x)}}{\frac{1}{f(y)}}dy} = \frac{f(y|x)}{\int \frac{\frac{f(x,y)}{f(x)}}{\frac{f(x,y)}{f(y)}}dy} = \frac{f(y|x)}{\int \frac{f(y|x)}{f(x|y)}dy}$$

Thus we can see that the joint $f(x,y)$ can be written in terms of the two conditions: $f(x|y)$ and $f(y|x)$.

2 Problem 2

We have that $Y \sim \chi^2_p$ and we have that $\mathbf{X} \sim MVN(0, \mathbf{I})$. This means that $X_1, X_2 \dots X_n \sim N(0,1)$ and we have the following:

$$f_y(y) = \frac{1}{2^{k/2}\Gamma(k/2)} y^{k/2-1} e^{-y/2}$$
$$f_{\mathbf{X}}(\mathbf{x}) = (2\pi)^{-k/2} e^{-1/2||\mathbf{x}||^2}$$

We need to perform the following transform: $\mathbf{Z} = \frac{\sqrt{Y}\mathbf{X}}{||\mathbf{X}||}$. This means that we have $Z_i = \frac{\sqrt{Y}X_i}{||\mathbf{X}||}$. We also define $W = \frac{\sqrt{Y}}{||X||}$. From the previous definition we have that $Z_i = WX_i$. We have the following inverse transformations: $X_i = \frac{Z_i}{W}$ and we have that $Y = W^2||\mathbf{X}||^2 = W^2\Sigma_{i=1}^n X_i^2 = W^2\Sigma_{i=1}^n \frac{W^2X_i^2}{W^2} = \Sigma_{i=1}^n W^2X_i^2 = \Sigma_{i=1}^n Z_i^2$. The jacobian is as follows:

$$J = \begin{bmatrix} \frac{\partial X_1}{\partial Z_1} & \frac{\partial X_1}{\partial Z_2} & \cdots & \frac{\partial X_1}{\partial Z_n} & \frac{\partial X_1}{\partial W} \\ \frac{\partial X_2}{\partial Z_1} & \frac{\partial X_2}{\partial Z_2} & \cdots & \frac{\partial X_2}{\partial Z_n} & \frac{\partial X_2}{\partial W} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \frac{\partial X_n}{\partial Z_1} & \frac{\partial X_n}{\partial Z_2} & \cdots & \frac{\partial X_n}{\partial Z_p} & \frac{\partial X_n}{\partial W} \\ \frac{\partial Y}{\partial Z_1} & \frac{\partial Y}{\partial Z_2} & \cdots & \frac{\partial Y}{\partial Z_n} & \frac{\partial Y}{\partial W} \end{bmatrix} = \begin{bmatrix} \frac{1}{W} & 0 & \cdots & 0 & \frac{-Z_1}{W^2} \\ 0 & \frac{1}{W} & \cdots & 0 & \frac{-Z_2}{W^2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{1}{W} & \frac{-Z_n}{W^2} \\ 2Z_1 & 2Z_2 & \cdots & 2Z_n & 0 \end{bmatrix}$$

If we take the determinant of this jacobian we have that $|J| = \frac{2}{W^{n+1}}||Z||^2$. We assume they are independent thus: $f_{W,Z}(w,z) = f_X(Z/W)f_Y(||Z||)|J|$. If we integrate out W we have that Z is a MVN distribution (this took many pages of scratch work).

3 Problem 3

See readme.txt file and code in /class/stat580/ashum/homework2/function_passing

4 Problem 4

See readme.txt file and code in /class/stat580/ashum/homework2/CCS_run

5 Problem 5

Instead of using $k+1$ for loops for all the summations my original idea was to use an array of bits or boolean variables. Each element in this array represents one of the X_i variables and is either 0 or 1. We would have a for loop to iterate from $i = 0$ to $i = 2^{k+1} - 1$, where $k+1$ is the same number of X_i variables. For each iteration we would convert i to a bit array (an array of bits corresponding to the binary representation of i). We could then iterate through this bit array for the values of X_i as we take the product in the likelihood. When we complete all $2^{k+1} - 1$ iterations this will try all combinations 0 and 1 for the X_is. However, the C language makes it quite difficult to create a bit array from an integer.

Instead we could convert the integer i in our for loop into binary representation. We could then check if the jth digit in the binary representation is 1 or 0 by using a second binary number that is all 0 with a single 1 in the jth position and using the AND operator on the two binary integers. If the resulting binary integer is zero then in our binary representation of integer i at position j is 0 and otherwise it will be non-zero. The shift operator comes into play when we want to check all binary digits of integer i. We would have a second binary number with all 0 and a 1 in the first position. We could iterate through by shifting the only non-zero bit and apply the AND operator with the binary representation of i; this would be equivalent to iterating through each position of a bit array as described above.

6 Problem 6

See readme.txt file and code in /class/stat580/ashum/homework2/power_iteration