Exercise 08: Spatial Smoothing at Scale

Laplacian Smoothing:

(A)

A is of the form $A = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}$ where $a_{ij} = \begin{cases} 1, \ if \ \exists\{i,j\} \\ 0, \ if \ \nexists\{i,j\} \end{cases}$.

W is of the form $W = \begin{bmatrix} w_{11} & 0 & \cdots & 0 \\ 0 & w_{12} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_{nn} \end{bmatrix}$ where $w_{ii} = \#$ of neighbours of vertex i.

$L = W - A = \begin{bmatrix} w_{11} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & w_{12} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & w_{nn} \end{bmatrix}$.

Therefore, l_{ii} is the # of neighbours of vertex i, and l_{il} is negative the # of adjacency.

$D = D_{m\times n} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \cdots & d_{mn} \end{bmatrix}$, where $d_{ij} = \begin{cases} 0, \ if \ \nexists \ adjacency \ for \ j \\ 1, \ if \ \exists\{j,k\} \ and \ j \leq k \\ -1 \ if \ \exists\{k,j\} \ and \ j > k \end{cases}$.

Therefore, $D^T D$ is a $m \times m$ matrix and the ij^{th} element of $D^T D$ is $dd_{ij} = \sum_{k=1}^{n} d_{k,i} d_{k,j}$.

If $i \neq j$, then $dd_{ij} = \begin{cases} 0, \ if \ \nexists\{i,j\} \\ -1, \ if \ \exists\{i,j\} \end{cases}$;

If $i = j$, then $dd_{ii} = \sum_{k=1}^{n} d_{k,i} d_{k,i} = \#$ of edges from/to (just for the sake of notation. There is no direction in the graph though.) node i.

It is easy to see that dd_{ij} is exactly the same as l_{ij}

(B)

The Laplacian smoothing problem becomes:

$$\underset{x \in \mathbb{R}^n}{minimize} \ \frac{1}{2}\|y - x\|_2^2 + \frac{\lambda}{2}\|Dx\|_2^2$$

Let $F(x) = \frac{1}{2}\|y-x\|_2^2 + \frac{\lambda}{2}\|Dx\|_2^2 = \frac{1}{2}(y-x)^T(y-x) + \frac{\lambda}{2}x^TD^TDx$

By FOC, $\frac{\partial}{\partial x}F = (x-y) + \lambda D^TDx = (1+\lambda D^TD)x - y = 0$

$\Rightarrow (1+\lambda L)\hat{x} = y$

(C)

Solve $C\hat{x} = b$, where $C = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$, $b = \begin{bmatrix} b_1 \\ b_1 \\ \vdots \\ b_n \end{bmatrix}$

Gauss-Seidel method:

$C = L_* + U$, where $L_* = \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ c_{21} & c_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{bmatrix}$, $U = \begin{bmatrix} 0 & c_{12} & \cdots & c_{1n} \\ 0 & 0 & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$

Therefore, the equation becomes $L_*x = b - Ux$, $x^{t+1} = L_*^{-1}(b - Ux^t)$

Step 1. Set initial value x^0, and $t = 1$;

Step 2. Repeat until converges:

(i) For $i = 1, \cdots, n$, set $x_i^{t+1} = \frac{1}{c_{ii}}(b_i - \sum_{j=1}^{i-1} c_{ij}x_j^{t+1} - \sum_{j=i+1}^{n} c_{ij}x_j^t)$

(ii) $t = t+1$

Step 3. Get final estimation $x^{t_{final}}$

The convergence properties of the GaussSeidel method are dependent on the matrix C.

Namely, the procedure is known to converge if either:

- C is symmetric positive-definite;

- C is strictly or irreducibly diagonally dominant.

The GaussSeidel method sometimes converges even if these conditions are not satisfied.

Jacobi Iterative method:

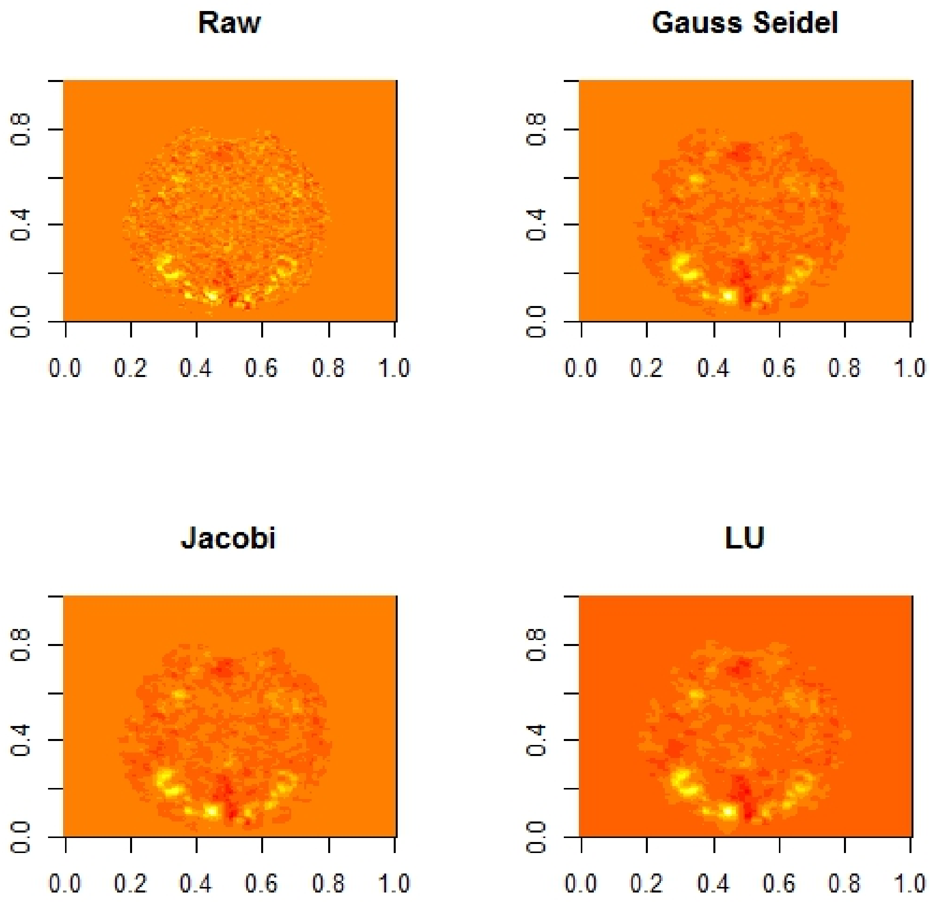
$C = D + R$, where $D = \begin{bmatrix} c_{11} & 0 & \cdots & 0 \\ 0 & c_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & c_{nn} \end{bmatrix}$, and $R = \begin{bmatrix} 0 & c_{12} & \cdots & c_{1n} \\ c_{21} & 0 & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & 0 \end{bmatrix}$

$Dx = b - Rx$, $x^{t+1} = D^{-1}(b - Rx^t)$

The standard convergence condition (for any iterative method) is when the spectral radius of the iteration matrix is less than 1.

A sufficient (but not necessary) condition for the method to converge is that the matrix A is strictly or irreducibly diagonally dominant.

Comparison:



The original problem is to minimize

$$\underset{x\in\mathbb{R}^n}{minimize}\frac{1}{2}\|y-x\|_2^2+\lambda|Dx|_1$$

Rewrite the problem as

$$\underset{x\in\mathbb{R}^n}{minimize}\frac{1}{2}\|y-x\|_2^2+\lambda|r|_1$$
$$\text{subject to } Dx=r$$

Rewrite the problem as $\underset{x,r}{minimize}\frac{1}{2}\sum_{i=1}^{n}\frac{\eta_i(y_i-x_i)^2}{2}+\lambda|r|_1+I_C(x,r)$,

where $I_C(x,r)$ takes the value 0 whenever $(x,r)\in C$, and ∞ otherwise. C is the convex set $C=x,r:Dx=r$.

Introduce 2 sets of slack variables (z for x, s for r). This yields another new problem:

$$\underset{x,z,r,s}{minimize}\frac{1}{2}\sum_{i=1}^{n}\frac{\eta_i(y_i-x_i)^2}{2}+\lambda|r|_1+I_C(z,s)$$
$$\text{subject to } x=z\in\mathbb{R}^n \;\&\; r=s\in\mathbb{R}^m$$

We now have a constrained optimization in four sets of primal variables x,z,r,s. Let u be the scaled dual variable corresponding to the constraint $x=z$, and let t be the scaled dual variable corresponding to the constraint $r=s$. We can write the augmented Lagrangian of problem in scaled form as

$L_x(x,z,r,s,t,u)=\frac{1}{2}\sum_{i=1}^{n}\frac{\eta_i(y_i-x_i)^2}{2}+\lambda|r|_1+I_C(z,s)+\frac{a}{2}\|x-z+u\|_2^2+\frac{a}{2}\|r-s+t\|_2^2$. Here a is the step-size parameter.

Algorithm:

Updating x:

$x^{t+1}=\underset{x}{argmin}\frac{1}{2}\sum_{i=1}^{n}\frac{\eta_i(y_i-x_i)^2}{2}+\frac{a}{2}\|x-z^t+u^t\|_2^2$.

This is separable in each component of $x_i^{t+1}=\frac{\eta_i y_i+a(z_i^t)-u_i^t}{\eta_i+a}$.

Updating r:

$r^{t+1}=\underset{r}{argmin}\lambda|r|_1+\frac{a}{2}\|r-s+t\|_2^2$.

This is also separable in each component, with minimum given by the soft-thresholding operator:

$r_j^{t+1}=S_{\lambda a}(s_j^t-t_j^t)$, and the soft thresholding operator is $S_a(x)=sign(x)|x-a|_+$

Updating (z,s):

The update in (z,s) must be done jointly. It is the only computationally demanding step of the algorithm. Specifically, we have

$(z^{t+1},s^{t+1})=\underset{(z,s)}{argmin}I_C(z,s)+\frac{a}{2}\|x-z+u\|_2^2+\frac{a}{2}\|r-s+t\|_2^2$

Equivalentely, $\underset{z,s}{minimize}\|z-w\|_2^2+\|Dz-v\|_2^2$

subject to $s = Dz$

Since s does not appear in the objective, we can just solve for z and then set $s = Dz$. The ordinary first-order optimality condition for z in the above optimization problem is $(I + D^T D)z^{t+1} = w + D^T v$, recalling that $w = x^{t+1} + u^t$ and $v = r^{t+1} + t^t$

Code for Laplacian Smoothing:

```
#### FMRI Data ####

fmri=read.csv( "C:/Users/Yuxin/Dropbox/Courses/2016 Fall/Stat Model for Big Data/Exercis

makeD2_sparse = function (dim1, dim2)  {
require(Matrix)
D1 = bandSparse(dim1 * dim2, m = dim1 * dim2, k = c(0, 1),
        diagonals = list(rep(-1, dim1 * dim2), rep(1, dim1 *
            dim2 - 1)))
D1 = D1[(seq(1, dim1 * dim2)%%dim1) != 0, ]
D2 = bandSparse(dim1 * dim2 - dim1, m = dim1 * dim2, k = c(0,
        dim1), diagonals = list(rep(-1, dim1 * dim2), rep(1,
        dim1 * dim2 - 1)))
return(rBind(D1, D2))
}

#### Gauss Seidel ####

GaussSeidel = function( x0=x0, b=y, C , maxiter, tol ){
x=x0
n= length(y)
t=1

L_star = tril(C)
U = triu(C, 1)

while(t<maxiter){

x.t = solve(L_star, b - U %*% x[,t])
```

```
x=cbind(x,x.t)
t=t+1

if( sum(abs( x[,t] - x[,t-1]) )<tol ){
break
}

}

return( list( x=x , iter=t, x.final=x[,t] ) )
}

n= length(y)
x0=matrix(0, nrow=n, ncol=1)

lambda = 1
C = Diagonal(n) + lambda * L
x0=matrix(0, nrow=n, ncol=1)

fit.GaussSeidel=GaussSeidel(x0=x0, b=y, C=C, maxiter=50, tol=1e-4 )
fit.GaussSeidel$iter

x.final.GaussSeidel=fit.GaussSeidel$x.final

x.Matrix.GaussSeidel = matrix(x.final.GaussSeidel, nrow = nrow(fmri))



#### Jacobi ####

Jacobi = function( x0=x0, b=y, C , maxiter, tol ){
x=x0
n= length(y)
```

```
t=1

D=Diagonal(x=diag(C))
R = C - Diagonal(x = diag(C))

while(t<maxiter){

x.t = solve(D, b - R %*% x[,t])

x=cbind(x,x.t)
t=t+1

if( sum(abs( x[,t] - x[,t-1]) )<tol ){
break
}

}

return( list( x=x , iter=t, x.final=x[,t] ) )
}
```