The following code is what I used to treat land-cover data as binary case: APB vs. Others. Since you are considering multinomial case, the calculation of probability on each site would be different.

There are four parameters in this algorithm: beta, v, b and Z.

1. beta should still have conjugacy even considering multi-categorical case.
2. v and b are used to define the relation among neighbors, and they might be diagonal matrixes in the multinomial case instead of scalars.
3. Z is the latent variable with mean beta in your case, and M-H algorithm should be used.

The codes in yellow are where you need to do some changes according to your set-up.

# 08/24/2011

# 4 categories-> 2categories: APB vs. Others, logit.

# only consider 1 parameter: mxpolypr + intercept

# get the DIC

# 08/26

# N=5000

# use abs() to get neighbor structure!

# starting from glm

# 08/30

# run 50000 loops!

library(mvtnorm)

N=50000 # Number of Loops!

eps=0.1 # TUNING 1 range of b

xi2=100 # prior of b

tau\_z=1.5 # TUNING 2 SD of Z

# setwd("C:/Users/JIN/Desktop/bin.mcmc")

land = read.csv("land.new.csv", header=T) # Read Data

# str(land)

I=1429

K=2

X=cbind(rep(1,I),land$mxpolypr)

colnames(X)=c('intercept','mxpolypr')

###################################

# Define Neighborhood by distance #

###################################

m\_X=matrix(rep(land$x,I),nrow=I)

m\_Y=matrix(rep(land$y,I),nrow=I)

m.abs\_dis=abs(m\_X-t(m\_X))+abs(m\_Y-t(m\_Y))

C=((m.abs\_dis>0)&(m.abs\_dis<850)) # sum(C)=5166

sum(C)

# m\_dis=((m\_X-t(m\_X))^2+(m\_Y-t(m\_Y))^2)^.5 # use ^2 instead of abs to calculate distance!!

# C=((m\_dis>0)&(m\_dis<850)) # sum(C)=5304

eigen.C=eigen(C)$values

b\_min=1/sort(eigen.C)[1]

b\_max=1/sort(eigen.C)[I]

Y=(land$cover=='APB')

###########################

# Matrices to put results #

###########################

beta=array(0,c(K,N))

v=rep(0,N)

b=rep(0,N)

Z=array(0,c(I,N))

I\_I=array(0,c(I,I))

diag(I\_I)=rep(1,I)

I\_K=array(0,c(K,K))

diag(I\_K)=rep(1,K)

#########

# Prior #

#########

sigma\_beta=100 # prior of beta

TT=I\_K/(sigma\_beta^2)

alpha20=1 # prior of v

gamm20=1

alpha2=alpha20+I/2 # posterior of v

tot.jump.b=tot.jump.Z=0 # count of jump in M-H algorithm

###################

# Starting values #

###################

fit=glm(Y~mxpolypr,data=land,family=binomial)

beta[,1]=fit$coeff

v[1]=1

b[1]=0

Z[,1]=log(fit$fitted/(1-fit$fitted))-0.1

set.seed(830)

date()

#########

# Loops #

#########

for(n in 2:N)

{

##########

# 1.beta #

##########

S=v[n-1]\*(I\_I-b[n-1]\*C)

A=t(X) %\*% S %\*% X + TT

B=t(X) %\*% S %\*% Z[,n-1]

Sigma\_beta=solve(A)

mu\_beta=Sigma\_beta%\*%B

beta[,n]=rmvnorm(1,mu\_beta,Sigma\_beta)

# the posterior of beta is normal dist.

##########

# 2.v #

##########

H=I\_I-b[n-1]\*C

mu= X %\*% beta[,n]

tmp=Z[,n-1] - mu

gamm2=gamm20 + 0.5\*t(tmp) %\*% H %\*% tmp

v[n]=rgamma(1, alpha2, rate=gamm2)

#######

# 3.b #

#######

b1=runif(1,b[n-1]-eps,b[n-1]+eps)

if((b1>b\_min)&(b1<b\_max)) {

H1=I\_I-b1 \* C

r=0.5 \* v[n] \* t(tmp) %\*% (H1-H) %\*% tmp +(b1^2-b[n-1]^2)/xi2

ratio1=det(H1)^0.5/(det(H)^0.5)\*exp(-r)

if(runif(1)<ratio1) {b[n]=b1; H=H1; jump=1} else {b[n]=b[n-1]; jump=0}

}

if((b1<=b\_min)|(b1>=b\_max)) {b[n]=b[n-1]; jump=0}

tot.jump.b=tot.jump.b+jump

##########

# 4.Z #

##########

Z[,n]=Z[,n-1]

u=v[n]\*b[n]

for(i in 1:I){

Z[i,n]=rnorm(1,Z[i,n-1],sd=tau\_z)

r=0.5 \* v[n] \* ((Z[i,n-1]-mu[i])^2-(Z[i,n]-mu[i])^2)

for(k in which(C[i,]!=0))

r=r + u \* (Z[i,n-1]-Z[i,n]) \* (Z[k,n]-mu[k])

pi1=exp(Z[i,n])/(1+exp(Z[i,n]))

pi0=exp(Z[i,n-1])/(1+exp(Z[i,n-1]))

ratio2=(pi1^Y[i]\*(1-pi1)^(1-Y[i]))/(pi0^Y[i]\*(1-pi0)^(1-Y[i])) \* exp(r)

if(runif(1)>ratio2) {Z[i,n]=Z[i,n-1]; jump=0} else {jump=1}

tot.jump.Z=tot.jump.Z+jump

}

}

date()

# acceptance ratio of M-H

tot.jump.b/N

tot.jump.Z/I/N

##########################

# TRACE PLOT & HISTOGRAM #

##########################

index=(N/5+1):N

# rownames(beta)=c('intercept','mxpolypr')

I4=sample(1:I,4)

jpeg(paste('0830.lc.apb.abs.lr1',N,eps,tau\_z,'TH.jpeg'),w=960,h=960)

par(mfrow=c(4,4))

# beta

for(k in 1:K){

plot(beta[k,],type='l',ylab=paste(colnames(X)[k]),main=paste('beta\_',k,sep=''),cex.main=2)

abline(v=N/5,col=2)

hist(beta[k,index],xlab=paste(colnames(X)[k]),freq=FALSE, main=paste('beta\_',k,': after burn',sep=''),cex.main=2)

}

plot(v,type='l',ylab='v',main='v',cex.main=2)

abline(v=N/5,col=2)

hist(v[index],xlab='v',freq=FALSE,main='v: after burn',cex.main=2)

plot(b,type='l',ylab='b',main='b',cex.main=2)

abline(v=N/5,col=2)

hist(b[index],xlab='b',freq=FALSE,main='b: after burn',cex.main=2)

# Z

for(i in I4){

plot(Z[i,],type='l',main=paste('Z\_',i,sep=''),cex.main=2)

abline(v=N/5,col=2)

hist(Z[i,index],freq=FALSE,main=paste('Z\_',i,': after burn',sep=''),cex.main=2)

}

dev.off()

########

# Grid #

########

###########################

# YY: real data as a grid #

###########################

aa=min(land$x)

bb=min(land$y)

uv=cbind(round((land$x-aa)/800,0)+2,round((land$y-bb)/790,0)+2)

YY=array(0,c(70,44))

YY[uv[land$cover!='APB',]]=1

YY[uv[land$cover=='APB',]]=2

############

# Estimate #

############

# Simulated result of w is too big, so we shrink it to 1/20

shrink2<-function(M) # for 2-d

{

a=dim(M)

b=length(a)

N=a[b]

a[b]=N/20

M\_s=array(0,a)

for(i in 1:a[b])

M\_s[,i]=M[,i\*20]

return(M\_s)

}

Zs=shrink2(Z)

M=N/20

index2=(1+M/5):M

PP1=exp(Zs[,index2])/(1+exp(Zs[,index2]))

YY1=matrix(rep(Y,M\*4/5),nrow=I)

S=-2 \* sum(log(PP1^YY1\*(1-PP1)^(1-YY1)))

D\_avg=S/(M\*4/5)

D\_avg

PP2=apply(PP1,1,median)

D\_est=-2 \* sum(log(PP2^Y\*(1-PP2)^(1-Y)))

D\_est

dic\_apb.LR1=2\*D\_avg-D\_est

dic\_apb.LR1 # DIC value

#######################################

# EE: estimated value as a grid #

# Estimate is based on majority rule! #

#######################################

EE=array(0,c(70,44))

EE[uv[PP2< .5,]]=1 # OTH

EE[uv[PP2>=.5,]]=2 # APB

EE1=EE2=array(0,c(70,44))

EE1[uv]=2 # background

EE2[uv]=PP2 # prob of APB

jpeg(paste('0830.lc.apb.abs.lr1',N,eps,tau\_z,'GR.jpeg'),w=960,h=720)

par(mfrow=c(2,2))

image(YY,col=gray(2:0/2),zlim=c(0,2),axes=FALSE,yaxt='n',xaxt='n',main='Land Cover',cex.main=1.5)

legend("topleft",c('OTH','APB'),pch=c(15,15),col=gray(1:0/2),pt.cex=1.5,cex=1.5)

image(EE,col=gray(2:0/2),zlim=c(0,2),axes=FALSE,yaxt='n',xaxt='n',main='Estimated',cex.main=1.5)

legend("topleft",c('OTH','APB'),pch=c(15,15),col=gray(1:0/2),pt.cex=1.5,cex=1.5)

image(EE1-(EE==YY),col=gray(2:0/2),zlim=c(0,2),axes=FALSE,yaxt='n',xaxt='n',main='Comparison',cex.main=1.5)

legend("topleft",c('Right','Wrong'),pch=c(15,15),col=gray(1:0/2),pt.cex=1.5,cex=1.5)

image(EE2,col=gray((32:0)/32),zlim=c(0,1),axes=FALSE,yaxt='n',xaxt='n',main='Prob of APB',cex.main=1.5)

dev.off()

sum(YY!=EE) # count of the wrong estimates.

tot.jump.b/N

tot.jump.Z/I/N

#####################################

# Write the results into .csv files #

#####################################

setwd('/scratch/jinc')

write.csv(t(signif(beta,4)), file = paste('0830.lc.apb.abs.lr1',N,eps,tau\_z,'beta.csv'))

write.csv(signif(v,4), file = paste('0830.lc.apb.abs.lr1',N,eps,tau\_z,'.v.csv'))

write.csv(signif(b,4), file = paste('0830.lc.apb.abs.lr1',N,eps,tau\_z,'.b.csv'))

write.csv(t(signif(Zs,4)), file = paste('0830.lc.apb.abs.lr1',N,eps,tau\_z,'.Zs.csv'))