Computer exercise INLA

Loading R-packages

Load the needed packages:

```r
> library(INLA)
> library(plotrix)
> library(geoR)
> library(fields)
> library(maps)
> library(spatstat)
```

You might need to install INLA

```r
> source("http://www.math.ntnu.no/inla/givemeINLA.R")
```

Loading the data

As an example we'll study the locations of 3605 trees in a tropical rain forest. The data comes with covariate data giving the elevation (altitude) and slope of elevation in the study region. The data could be analysed as a spatial-point process, but we'll simplify matters and compute the number of trees in each grid cell. The data is then modelled using a Poisson-process with spatially varying mean.

```r
> #Load and examine the data
> data(bei)
> plot(bei)
> #extract elevation and gradient
> elevation <- bei.extra$elev$v
> gradient <- bei.extra$grad$v
> #compute counts in each grid-cell
> counts <- matrix(0,dim(gradient)[1],dim(gradient)[2])
> #size of each grid cell
> dx <- (bei.extra$elev$xcol[2] - bei.extra$elev$xcol[1])/2
> dy <- (bei.extra$elev$yrow[2] - bei.extra$elev$yrow[1])/2
> #loop over the grid cells, and count points
> for(i in 1:dim(counts)[1]){  
    Y <- bei.extra$grad$yrow[i]
    for(j in 1:dim(counts)[2]){  
        X <- bei.extra$grad$xcol[j]
        counts[i,j] <- sum((X-dx)<bei$x & bei$x<=(X+dy) & (Y-dy)<bei$y & bei$y<=(Y+dx))
    }
}
```

Let's plot the resulting gridded data,

```r
> par(mfrow=c(2,2))
> image.plot(elevation)
> image.plot(gradient)
> image.plot(counts)
```

and collect the data into a suitable data-frame

```r
> bei <- data.frame(counts=c(counts),  
                    x=rep(bei.extra$elev$xcol, each=length(bei.extra$elev$yrow)),  
                    y=rep(bei.extra$elev$yrow, length(bei.extra$elev$xcol)),  
                    elevation=c(elevation), gradient=c(gradient))
```
Triangulation
The first step in creating a Gaussian Markov random field solution to the Stochastic Partial Differential Equation, as described in the lecture notes, is to create a triangular mesh over the observation locations. We could create a mesh for the lattice

```r
def mesh.lattice <- inla.mesh.lattice(x=bei.extra$elev$xcol, y=bei.extra$elev$yrow)
def mesh.lattice <- inla.mesh.create(lattice=mesh.lattice)
plot(mesh.lattice)
summary(mesh.lattice)
```

However this mesh is much denser than needed, instead we compute a mesh with a suitable internal triangulation (max edge length 50) and coarser extension.

```r
def loc <- as.matrix(bei[,c("x","y")])
def mesh <- inla.mesh.create.helper(points.domain=loc, offset = c(-0.05, -0.2), n=c(8,16), max.edge = c(25,150))

summary(mesh)
##plot
plot(mesh)
##add the points for reference (note the possible aliasing effect)
points(loc[,1], loc[,2], col="red", pch=19, cex=.1)
```

Precision matrices
Given the mesh the next step is to compute the matrices needed to create the precision matrix \(Q\), and the observation matrix needed to link the mesh to our observations locations.

```r
def spde <- inla.spde2.matern(mesh, alpha=2)
def A <- inla.spde.make.A(mesh, loc)
```

The resulting matrices are sparse.

```r
image( spde$param.inla$M0 )
image( spde$param.inla$M1 )
image( spde$param.inla$M2 )
image(A)
##or only the first 1000 observations
image(A[1:1000,])
```

In addition to these we also need an index vector for the latent field

```r
def spde.idx <- inla.spde.make.index("spatial", n.mesh=mesh$n)
```

Setting up the model
The next step is to create a list of effects and an INLA-stack object containing the relevant components of the model. For numerical reasons we select to put the intercept with the field; the second component of the effects contains all possible covariates. The stack object combines the observations, observations matrices, effects, and a tag which can be used to identify components of the posterior.

```r
def effects <- list(c(list(intercept=rep(1,mesh$n)), spde.idx),
                   covar=bei[,c("x","y","elevation","gradient")])
def stack.obs <- inla.stack(data=list(Y=bei$counts), A=list(A, 1),
effects=effects, tag="obs")
```

The stack must contain one observation matrix for each effect. Here the 1 indicates an identity matrix linking the covariates (second element in effects) with the observations. Predictions at unobserved locations can be added by creating a stack without observations, i.e. \(Y = NA\).
The model

The model defined in stack.obs above can be written as

\[ y_i \sim \text{Po}(e^{z_i}) \quad Z = Aw + IB\beta w \quad \in \mathcal{N}(0, Q)\beta \in \mathcal{N}(0, I \cdot 10^2) \]

Thus we have Poisson observations, \(y_i\); with intensity, \(z_i\); determined by a latent field, \(w\); linked to the observations location through an observation matrix, \(A\); and with a mean consisting of covariates, \(B\); and with a Normal-prior for the regression coefficients. Strictly speaking the model only defines things up to \(z_i\), the type of observations are defined in the call to inla in a few paragraphs.

In principal the latent field is expanded to

\[ Z = [A \quad B] \begin{bmatrix} w \\ \beta \end{bmatrix} \]
\[ \begin{bmatrix} w \\ \beta \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} Q_{\text{spde}} & 0 \\ 0 & I \cdot 10^{-2} \end{bmatrix}^{-1} \right) \]

Priors

The default INLA priors can be extracted, and changing the values is straightforward. For the poission observations we have

```r
> hyper.obs <- inla.models()$likelihood$poisson$hyper
> hyper.obs
```

Which is rather uninteresting. The prior for Gaussian observations has a little more to offer

```r
> hyper.gaussian <- inla.models()$likelihood$gaussian$hyper
> str(hyper.gaussian)
```

Even the spatial model (the spde-object) comes with a default prior; the range parameter has been chosen based on the coordinates of the mesh.

```r
> spde$param.inla$theta.mu
> spde$param.inla$theta.Q
```

Finally the precision of the regression coefficients, \( \beta \), can be altered from the predetermined values through the control.fixed option in the call to inla, see

```r
> ?control.fixed
```

Running INLA

Having defined the model, data structure, initial values, and prior distributions we are ready to estimate the model. We try both a simple, and a more complex mean.

In the following the model is defined using the standard \( \sim \) notation in R with the names refering to elements in stack.obs. family="poisson" tells INLA that our data should be considered as Poisson observations of the latent field. Other possible observation models are:

```r
> names(inla.models()$likelihood)
```

Finally control.family allows us alter the priors of the observation likelihood.

DO NOT RUN!!!

Estimation of the two models.

```r
> res1 <- inla(Y ~ -1 + intercept + f(spatial, model=spde),
family="poisson", data = inla.stack.data(stack.obs),
control.family = list(hyper = hyper.obs),
control.predictor = list(A=inla.stack.A(stack.obs),
```
compute=TRUE),
control.results=list(return.marginals.random=FALSE,
return.marginals.predictor=FALSE), verbose=TRUE)
> res2 <- inla(Y ~ -1 + intercept + elevation + gradient + x + y +
+f(spatial, model=spde), family="poisson",
data = inla.stack.data(stack.obs),
control.family = list(hyper = hyper.obs),
control.predictor = list(A=inla.stack.A(stack.obs),
compute=TRUE),
control.results=list(return.marginals.random=FALSE,
return.marginals.predictor=FALSE), verbose=TRUE)

Run this instead

Load precompute values instead.
> load("INLAres.Rdata")

We extract parameters for the latent field using
> spde.res1 <- inla.spde2.result(res1, "spatial", spde)
> spde.res2 <- inla.spde2.result(res2, "spatial", spde)

If the option return.marginals.random=TRUE had been chosen above these
structures would also include posteriors for the field-parameters, rather than
only summaries.

Analysing INLA output

Estimated parameters

As a first step in analysing the output we compare the estimated regression
parameters for the two models,
> res1$summary.fixed
> res2$summary.fixed

and for the hyper-parameters of the latent field.
> res1$summary.hyperpar
> res2$summary.hyperpar

Or the equivalent range and variances
> spde.res1$summary.log.range.nominal
> spde.res2$summary.log.range.nominal
> spde.res1$summary.log.variance.nominal
> spde.res2$summary.log.variance.nominal

We can also plot the posterior distributions of the hyper-parameters, along with
their priors (in red)
> par(mfcol=c(2,1))
> theta.mu <- spde$param.inla$theta.mu
> theta.Q <- spde$param.inla$theta.Q
> for(i in 1:2){
##posterior
plot(res1$marginals.hyperpar[[i]], type="l",
main=names(res1$marginals.hyperpar)[i])
##prior
x <- res1$marginals.hyperpar[[i]][,1]
lines(x, dnorm(x, theta.mu[i], sqrt(1/theta.Q[i,i])),
col="red")
##and for the second model
lines(res2$marginals.hyperpar[[i]], col="blue")
}

Marginals for the regression parameters can be found in
> str(res2$marginals.fixed)

Latent field

Let's first plot the observations and the mean component of the second field

> par(mfrow=c(2,1))
> image.plot( as.image(bei$counts, x=bei[,c("x","y")]),asp=1 )
> ##compute the mean, and plot
> B <- as.matrix( cbind(1,bei[,rownames(res2$summary.fixed)[-1]]) )
> mu <- B %*% res2$summary.fixed[,"mean"]
> image.plot( as.image(mu, x=bei[,c("x","y")]),asp=1 )

To plot the GMRF component of the fields we need to define a projector from the field to a suitable grid that can be used for plotting

> mesh.projector <- inla.mesh.projector(mesh, dims=c(200,200))

We can now plot the expectation and standard deviation of the field component

> ##study component
> str(res1$summary.random)
> ##extract mean and sd
> EZ.1 <- res1$summary.random$spatial[,"mean"]
> EZ.2 <- res2$summary.random$spatial[,"mean"]
> sd.1 <- res1$summary.random$spatial[,"sd"]
> sd.2 <- res2$summary.random$spatial[,"sd"]
> ##plot the fields
> par( mfrow=c(2,2) )
> image.plot(mesh.projector$x, mesh.projector$y,
> inla.mesh.project(mesh.projector, EZ.1),
> main="EZ.1")
> image.plot(mesh.projector$x, mesh.projector$y,
> inla.mesh.project(mesh.projector, EZ.2),
> main="EZ.2")
> image.plot(mesh.projector$x, mesh.projector$y,
> inla.mesh.project(mesh.projector, sd.1),
> main="sd.1")
> image.plot(mesh.projector$x, mesh.projector$y,
> inla.mesh.project(mesh.projector, sd.2),
> main="sd.2")
> ##add the mesh for reference to one of the plots
> plot(mesh,add=TRUE,col="grey")

A better alternative is to limit the plotting to the same are as the observations

> image.plot(mesh.projector$x, mesh.projector$y,
> inla.mesh.project(mesh.projector, EZ.1),
> xlim=range(bei$x), ylim=range(bei$y))

We have now plotted the mean component and the random component of the latent field, we would also like to plot the full latent field.

Predictions for the full field are contained in

> str(res1$summary.linear.predictor)

with indecies for the observed points given by

> I <- inla.stack.index(stack.obs, "obs")
> str(I)
> ##and extract the relevant components
> E1 <- res1$summary.linear.predictor[I$data,]
> E2 <- res2$summary.linear.predictor[I$data,]

Let's plot the field and uncertainties for the two fields
> par(mfrow=c(3,3))
> zlim <- range( c(E1[,"mean"],E2[,"mean"]))
> image.plot( as.image(E1[,"mean"], x=bei[,c("x","y")]),asp=1,
> xlab=range(bei$x), ylab=range(bei$y),
> zlim=zlim, main="EZ first field")
> image.plot( as.image(E2[,"mean"], x=bei[,c("x","y")]),asp=1,
> xlab=range(bei$x), ylab=range(bei$y),
> zlim=zlim, main="EZ second field")
> image.plot( as.image(bei$counts, x=bei[,c("x","y")]), asp=1,
> xlab=range(bei$x), ylab=range(bei$y),
> main="Observations")
> ##just the regression part
> image.plot( as.image(rep(res1$summary.fixed[,"mean"],dim(bei)[1]),
> x=bei[,c("x","y")]), asp=1,
> xlab=range(bei$x), ylab=range(bei$y),
> zlim=zlim, main="Mean first field")
> image.plot( as.image(mu, x=bei[,c("x","y")]), asp=1,
> xlab=range(bei$x), ylab=range(bei$y),
> zlim=zlim, main="Mean second field")
> plot(0,0,type="n")
> ##the uncertainties
> zlimV <- range( c(E1[,"sd"],E2[,"sd"]) )
> image.plot( as.image(E1[,"sd"], x=bei[,c("x","y")]),asp=1,
> xlab=range(bei$x), ylab=range(bei$y),
> zlim=zlimV, main="VZ first field")
> image.plot( as.image(E2[,"sd"], x=bei[,c("x","y")]),asp=1,
> xlab=range(bei$x), ylab=range(bei$y),
> zlim=zlimV, main="VZ second field")

Comparing the covariance to Matérn

We could also compare dependence in the SPDE model to those of the Matérn covariance we’re approximating.

First we compute the precision matrix given the estimated parameters

```r
> Q <- inla.spde2.precision(spde, res2$summary.hyperpar[,"mean"])
```

and study the sparsity

```r
> image(Q)
```

Inverting the precision matrix gives the covariance matrix, and correlation matrix (this takes several seconds)

```r
> S <- as.matrix(solve(Q))
> S <- diag(1/sqrt(diag(S))) %*% S %*% diag(1/sqrt(diag(S)))
> ##we also need the distance matrix
> D <- as.matrix(dist(mesh$loc[,1:2]))
```

We now need the covariance function and variograms for the SPDE model. We obtain the approximate posterior means for the “variance” $\tau$ and “range” $\kappa$.

```r
> tau <- exp( spde.res2$summary.log.tau[,"mean"] )
> kappa <- exp( spde.res2$summary.log.kappa[,"mean"] )
```

We also compute the theoretical Matérn correlation function

```r
> d <- seq(0, max(D), len=1e3)
> S.theory <- (d+kappa)*besselK(d+kappa,1)
> S.theory[1] <- 1
```

and compare the two

```r
> par(mfrow=c(2,1))
> ##using only a few interior points
```
> plot(D[,1:10], as.matrix(S[,1:10]), pch=19, cex=.1, xlab="dist", ylab="")
> lines(d, S.theory, col=2)
> # or only the edge points
> I <- unique(c(mesh$segm$bnd$idx))
> plot(D[,I], as.matrix(S[,I]), pch=19, cex=.1, xlab="dist", ylab="")
> lines(d, S.theory, col=2)

The rather larger spread around the theoretical function in the second plot is due to edge effects and the mesh size being reasonable large compared to the range. The first plot gives the correlation between 10 central points and the rest of the field, reducing the edge effects.

The end!