Spatial Statistics with Image Analysis
Lecture 12

Johan Lindström

December 08, 2017

Lecture L12

Computer exercise 5

General MRFs:
  Integer Valued Fields
  Pseudo-likelihood

Pixel classification

Markov Chain Monte Carlo
  Convergence

Project 3
  Corrupt Pixels
  Space-Time Data

Classification
False colour composite (or IR-colour) images are used to pick out healthy vegetation (chlorophyll).

University of Wisconsin-Madison Campus, artificial turf on the football field.


Integer valued fields — Joint distribution

- The global probability function for the simple integer field model is:

$$p(x) \propto \exp \left( \sum_{i=1}^{N} x_i + \frac{1}{2} \sum_{i=1}^{N} \sum_{j \in N_i} \beta \mathbb{1}(x_j = x_i) \right)$$

we could consider different neighbourhood strengths for different classes (i.e. \( \beta_i \)).

- The conditional distribution for a pixel \( x_i \) given the rest of the field is

$$p(x_i | x_j, j \neq i) = \frac{\exp \left( x_i + \beta \sum_{j \in N_i} \mathbb{1}(x_j = x_i) \right)}{\sum_{k=1}^{K} \exp \left( \alpha_k + \beta \sum_{j \in N_i} \mathbb{1}(x_j = k) \right)}.$$ 

Pseudo-likelihood estimation

For the estimation the set of points, \( I \), over which we compute the approximate likelihood

$$\log PL_z(\alpha, \beta) = \sum_{k} \alpha_k \sum_{i \in I} z_{ik} + \beta \sum_{k} \sum_{i \in I} z_{ik} f_{ik}$$

$$- \sum_{i \in I} \log \left( \sum_{k} \exp(\alpha_k + \beta f_{ik}) \right)$$

can be selected as either a set of conditionally independent points leading to one estimate for each set; or as all points in the field.
Pseudo-likelihood estimation

- Conditionally independent points:
  - Called coding-method.
  - Final estimate obtained by averaging each individual estimate.
  - Lower bias.

- All points:
  - Called pseudo-likelihood.
  - Nice asymptotic properties (consistent, Normal).
  - Lower variance.

Bayesian hierarchical modelling (BHM)

A hierarchical model is constructed by systematically considering components/features of the data, and how/why these features arise.

Bayesian hierarchical modelling

- A Bayesian hierarchical model typically consists of (at least)
- Data model, \( p(y|x, \theta) \): Describing how observations arise given the latent variables \( x \) and parameters \( \theta \).
- Latent model, \( p(x|\theta) \): Describing how the latent variables (reality) behaves, given parameters.
- Parameters, \( p(\theta) \): Describing our, sometimes vague, prior knowledge of the parameters.

Bayesian hierarchical model — So far

So far we have had:

- Kriging: Latent Gaussian-field with Gaussian observations.
- GMRF: Latent GMRF (i.e. Gaussian) with Gaussian or Poisson observations (in general non-Gaussian).
- Classification: Independent latent “field”, and any distribution (Gaussian) for the observations.

The remaining case is a latent DMRF (discrete MRF) with any distribution (Gaussian) for the observations. This will result in a classifier where neighbouring pixels tend to belong to the same class.
Object classification as a BHM

Treating the classification, with spatial dependence, as a BHM we have

Data model, \( p(y|x, \theta) \): Distribution of a pixel given its class belonging \( x \) and parameters \( \{\mu_k, \Sigma_k\}_{k=1}^K \).

\[
[y_i | x_i = k] \sim \mathcal{N} \left( \mu_k, \Sigma_k \right).
\]

Latent model, \( p(x|\theta) \): Describing how common and spatially contiguous each class is.

\[
p \left( x_i | x_j, j \in N_i, \theta \right) \propto \exp \left( \alpha x_i + \beta f_i x_i \right).
\]

Parameters, \( p(\theta) \): Describing our vague prior knowledge of the parameters.

Posterior fields — DMRF

Model with general observations and latent DMRF,

\[
p \left( y_i | x_i, \theta \right), \quad p \left( x_i | x_j, j \in N_i, \theta \right) \propto \exp \left( \alpha x_i + \beta f_i x_i \right).
\]

The posterior becomes a non-stationary DMRF,

\[
P \left( x_i | x_j, j \in N_i, y, \theta \right) \propto p \left( y_i | x_i, \theta \right) \exp \left( \alpha x_i + \beta f_i x_i \right)
\]

\[
= \exp \left( \alpha x_i + \log p(y_i|x_i, \theta) + \beta f_i x_i \right).
\]

Pixel classification using a DMRF

We now have that the posterior for the classification is a non-stationary DMRF with pixel dependents \( \tilde{\alpha}_{ik} \) which includes the data log-likelihood:

\[
\tilde{\alpha}_{ik} = \alpha_k + \log p(y_i | x_i = k, \theta)
\]

Inference for the entire model is now obtained from a Gibbs-algorithm that draws samples from:

1. The posterior DMRF using \texttt{mrf gaussian post.m} and \texttt{mrf sim.m}.
2. The class parameters \texttt{gibbs mu Sigma.m}.
3. The parameters of the DMRF \texttt{gibbs alpha beta.m}.
Sampling from $\alpha$ and $\beta$

Estimates of the parameters ($\alpha$ and $\beta$) for the DMRF are given by the log-pseudo-likelihood

$$\log PL_z(\alpha, \beta) = \sum_k \alpha_k \sum_i z_{ik} + \beta \sum_k \sum_i z_{ik} f_{ik} - \sum_i \log \left( \sum_k \exp(\alpha_k + \beta f_{ik}) \right)$$

To obtain reasonable samples of $\alpha$, $\beta$ we need to sample from this distribution wrt. $\alpha$, $\beta$.

This distribution is hard to sample from and we use a random-walk Metropolis-Hastings algorithm.

Sampling from complex distributions

We have previously introduced the Gibbs sampler to sample from complex distributions by sequentially sampling from the simpler conditional distributions.

Metropolis-Hastings is an alternative when the conditional distributions are hard (impossible) to sample from.

Metropolis-Hasting

Given a target density $p(x)$ from which we want to sample and a proposal distribution $q(x^{\text{new}}|x^{\text{old}})$. We construct a Markov chain $\{x^{(0)}, x^{(1)}, \ldots, x^{(T)}\}$ has stationary distribution $p(x)$ by:

1. Given the current state $x^{(t)}$, draw a new sample $x^* \in q(x^{(t+1)}|x^{(t)})$ from the proposal distribution.
2. Compute the acceptance probability

$$\alpha(x^{(t)}, x^*) = \min \left( 1, \frac{p(x^*) q(x^{(t)}|x^*)}{p(x^{(t)}) q(x^{(t)}|x^{(t)})} \right).$$

3. Set the next value in the chain to $x^{(t+1)} = x^*$ with probability $\alpha$ and to $x^{(t+1)} = x^{(t)}$ with probability $1 - \alpha$.
4. Repeat from 1.
A look at $\alpha(x^{(t)}, x^*)$

First ignore the proposal distribution $q(x^*|x^{(t)}$):

- $p(x^*)/p(x^{(t)})$ says accept (keep) the proposed value, $x^*$ if it is "better" than the old value $x^{(t)}$.
- If the new value is "worse" than the old accept it with a probability proportional to how much worse.

A word on notation:

The proposal distribution $q(x^{\text{new}}|x^{\text{old}})$ is also be called the proposal kernel $q(x^{\text{old}}, x^{\text{new}})$ (Note the order!).

However, some states may be easier to reach than others, $q(x^*|x^{(t)})$ compensats for this:

- If it is easy to reach $x^*$ from $x^{(t)}$, the denominator $q(x^*|x^{(t)})$ will reduce the acceptance probability.
- If it is easy to get back to $x^{(t)}$ from $x^*$, the nominator $q(x^{(t)}|x^*)$ will increase the acceptance probability.

Convergence — Burn in

Since the chain is started at an arbitrary point it needs to converge to the stationary distribution before we can use the samples.

The time for convergence is determined by looking at the chain (c.f. the Gibbs sampler for a DMRF) and then discarding the initial samples as Burn in.
Convergence diagnostics (cont.)

A simulated field (4-neighbours, $\alpha = 0, \beta = 1$) and the corresponding trajectory of the log-likelihood:

Symmetric proposal — Random-Walk

Select the proposal kernel so that

$$q(x^*|x(t)) = q(x(t)|x^*)$$

- The acceptance probability reduces to

$$\alpha(x(t), x^*) = \min \left(1, \frac{p(x^*)}{p(x(t))} \right).$$

- Commonly this is $x^* = x(t) + \varepsilon$ with $\varepsilon \in \mathcal{N}(0, \sigma^2)$.  

Selecting $\sigma^2$

The mixing depends on the proposal variance $\sigma^2$

- Good acceptance rate is around 30%.
- Smaller values of $\sigma^2$ increases the acceptance rate.
- Larger values of $\sigma^2$ decreases the acceptance rate.
- Bad mixing leads to various problems, including lack of convergence.
- $\sigma^2$ needs to be choosen “suitably” (tuning).

This is not an issue for the Gibbs-sampler since we “accept” all proposals.
Mixing

Mixing for three chains with different proposal variance $\sigma^2$. Good mixing. Too large $\sigma^2$. Too small $\sigma^2$.

Corrupt Pixels — A model

Pixels have been randomly replaced by incorrect values. A reasonable model, assuming an image with pixel values $y \in [0, 1]$, is:

Data model:

$$y(s_i)|x, z \in \begin{cases} 
N(x(s_i), \sigma^2) & \text{if } z(s_i) = 0 \\
U(0, 1) & \text{if } z(s_i) = 1 
\end{cases}$$

Latent model I: for the image

$$x|\kappa, \tau \in N(0, Q^{-1}(\kappa, \tau))$$

Latent model II: for independent bad pixels

$$p(z_i = k|p_c) = \begin{cases} 
p_c & \text{if } k = 0 \\
1 - p_c & \text{if } k = 1 
\end{cases}$$

Corrupt Pixels — Estimation

To estimate the model you will be using Gibbs sampling

1. Write down the full posterior for parameters $(\theta = \{\sigma^2, \kappa, \tau, p_c\})$ and latent fields $(x, z)$:

$$p(x, z, \theta|y) \propto p(y|x, z, \theta) p(x|\theta) p(z|\theta)$$

$$= p(y|x, z, \sigma^2) p(x|\kappa, \tau) p(z|p_c)$$

2. Compute the conditional posteriors

- $p(x|z, \theta, y)$ — Reconstruction in a GMRF
- $p(z|x, \theta, y)$ — Classification
- $p(\sigma^2|x, z, y)$ — Inverse-Gamma
- $p(x|z, y)$ — Gamma
- $p(p_c|x, z, y)$ — Beta (e.g. Dirichlet)

3. Construct a Gibbs sampler by alternating samples from
the above conditional distributions.
Corrupt Pixels — Comments

- We will keep \( k \) fixed. It can be sampled jointly with \( \tau \) using a random-walk Metropolis-Hastings.
- You might have to expand the field from \( x \) to \( \tilde{x} = [x^T \beta^T]^T \) to include a non-zero mean.
- Reordering should only be needed for the sampling step

\[
p(x|z, \theta, y)
\]

where the sparsity of \( R_{x|y} \) may matter.

Corrupt Pixels — Selected Posteriors

- \( x \) and \( z \) are latent fields giving the true image, and the bad pixels respectively.
- The posterior for a GMRF was given in Lecture 7:

\[
x \mid y, z, \theta \in N \left( \mu_{x|y}(z, \theta), Q_{x|y}(z, \theta) \right).
\]

where \( \mu_{x|y} \) and \( Q_{x|y} \) depend on \( z \) through the observation matrix \( A(z) \).
- \( z \mid y, x \), is a Bayesian classification problem (see Lecture 9)

\[
p(z_i = 0 | y_i, x_i, \theta) = \frac{p(y_i|z_i = 0, x_i, \theta) p(z_i = 0)}{\sum_k p(y_i|z_i = k, x_i, \theta) p(z_i = k)}
= \frac{p_N(y_i|x_i, \theta) p_c}{p_N(y_i|x_i, \theta) p_c + 1 \cdot (1 - p_c)}
\]

Corrupt Pixels — Selected Posteriors

For the GMRF model we have, given a sample of the field \( x \),

\[
p(\tau|x, k) \propto |Q(k, \tau)|^{1/2} \exp \left( -\frac{1}{2} \tau^T Q(k, \tau) x \right).
\]

With

\[
Q(k, \tau) = \tau Q_0(k).
\]

The density becomes

\[
p(\tau|x, k) \propto N^{1/2} |Q_0(k)|^{1/2} \exp \left( -\frac{\tau^T Q_0(k)x}{2} \right)
\approx N^{1/2 + 1 - 1} \exp \left( -\frac{\tau^T Q_0(k)x}{2} \right)
\]
**Space-Time Data — A model**

The fMRI data used for the classification project can also be seen as a spatio-temporal dataset

\[ y(s, t) = \sum_k \beta_k(s)X_k(t) + \epsilon(s, t) \]

\[ \epsilon(s, t) \in N \left( 0, \tau^{-1}(s) \right) \]

\[ \beta_k \in N \left( 0, \tau^{-1}_{\text{0}}{Q}_0^{-1} \right) . \]

Here \( X_k(t) \) are known temporal functions, marking the active blocks, or time periods, of the fMRI experiment.

**The Data**

The data consists of fMRI data arranged in a \( 87 \times 102 \times 160 \) array (i.e. \( 87 \times 102 \) pixels and 160 time points). Column stacking the image gives a \( 8874 \times 160 \) matrix where each row represents one pixel.

A second column stacking gives a vector as:

\[
\begin{bmatrix}
  y(s_1, t_1) \\
y(s_2, t_1) \\
  \vdots \\
y(s_n, t_1)
\end{bmatrix}
\begin{bmatrix}
y(s_1, t_2) \\
y(s_2, t_2) \\
  \vdots \\
y(s_n, t_2)
\end{bmatrix}
\begin{bmatrix}
y(s_1, t_T) \\
y(s_2, t_T) \\
  \vdots \\
y(s_n, t_T)
\end{bmatrix}
\rightarrow Y =
\begin{bmatrix}
y(s_1, t_1) \\
y(s_2, t_1) \\
  \vdots \\
y(s_n, t_1)
\end{bmatrix}
\begin{bmatrix}
y(s_1, t_2) \\
y(s_2, t_2) \\
  \vdots \\
y(s_n, t_2)
\end{bmatrix}
\begin{bmatrix}
y(s_1, t_T) \\
y(s_2, t_T) \\
  \vdots \\
y(s_n, t_T)
\end{bmatrix}
\]

**The Model — Matrix Form**

Given the column stacking we can write the model as

\[
\begin{bmatrix}
y(s_1, t_1) \\
y(s_2, t_1) \\
  \vdots \\
y(s_n, t_1)
\end{bmatrix}
\begin{bmatrix}
y(s_1, t_2) \\
y(s_2, t_2) \\
  \vdots \\
y(s_n, t_2)
\end{bmatrix}
\begin{bmatrix}
y(s_1, t_T) \\
y(s_2, t_T) \\
  \vdots \\
y(s_n, t_T)
\end{bmatrix}
= \sum_k \beta_k(s_1)X_k(t_1) + \sum_k \beta_k(s_2)X_k(t_1) + \sum_k \beta_k(s_n)X_k(t_1) + \epsilon(s_1, t_1)
\]

\[
+ \sum_k \beta_k(s_1)X_k(t_2) + \sum_k \beta_k(s_2)X_k(t_2) + \sum_k \beta_k(s_n)X_k(t_2) + \epsilon(s_1, t_2)
\]

\[
+ \sum_k \beta_k(s_1)X_k(t_T) + \sum_k \beta_k(s_2)X_k(t_T) + \sum_k \beta_k(s_n)X_k(t_T) + \epsilon(s_1, t_T)
\]
The Model — Matrix Form

For a $2 \times 2$ field with 2 time points we have:

$$
\begin{bmatrix}
    y(s_1, t_1) \\
    y(s_2, t_1) \\
    y(s_1, t_2) \\
    y(s_2, t_2)
\end{bmatrix} =
\begin{bmatrix}
    \sum_k \beta_k(s_1) X_k(t_1) \\
    \sum_k \beta_k(s_2) X_k(t_1) \\
    \sum_k \beta_k(s_1) X_k(t_2) \\
    \sum_k \beta_k(s_2) X_k(t_2)
\end{bmatrix}
+ \begin{bmatrix}
    \varepsilon(s_1, t_1) \\
    \varepsilon(s_2, t_1) \\
    \varepsilon(s_1, t_2) \\
    \varepsilon(s_2, t_2)
\end{bmatrix}
$$

The Model — Kronecker product

Rewriting the sum over $k$ as a matrix product (think regression) we have:

$$
\begin{bmatrix}
    y(s_1, t_1) \\
    y(s_2, t_1) \\
    y(s_1, t_2) \\
    y(s_2, t_2)
\end{bmatrix} =
\begin{bmatrix}
    X(s_1) 0 0 0 0 0 0 0 \\
    0 X(s_2) 0 0 0 0 0 0 \\
    0 0 X(s_1) 0 0 0 0 0 \\
    0 0 0 X(s_2) 0 0 0 0
\end{bmatrix}
\begin{bmatrix}
    \beta_1(s_1) \\
    \beta_2(s_1) \\
    \beta_1(s_2) \\
    \beta_2(s_2)
\end{bmatrix} +
\begin{bmatrix}
    \varepsilon(s_1, t_1) \\
    \varepsilon(s_2, t_1) \\
    \varepsilon(s_1, t_2) \\
    \varepsilon(s_2, t_2)
\end{bmatrix}
$$

Which, using kronecker products becomes

$$
Y = (X \otimes I_{n \times n}) \beta + \varepsilon
$$

The Model — Precision matrices

Noting that the $\beta$-fields are stacked after each other in $\beta$ we have that

$$
\beta = \begin{bmatrix}
    \beta_1 \\
    \beta_2 \\
    \beta_3
\end{bmatrix} \in \mathbb{N}
\begin{bmatrix}
    \tau_{0,1} & 0 & 0 \\
    0 & \tau_{0,2} & 0 \\
    0 & 0 & \tau_{0,3}
\end{bmatrix} \otimes Q_0^{-1}
$$

where

$$
Q_0 = \begin{cases}
    G & \text{if CAR} \\
    G^T G & \text{if SAR}
\end{cases}
$$
The Model — Observational Precision

Noting that the $\varepsilon$-fields are stacked in the same order as $Y$ and that we would like different variances for each location we have

$$\varepsilon(t,s) \in N \left( 0, \tau^{-1}_\varepsilon(s) \right)$$

or in vector form

$$\varepsilon = \begin{bmatrix} \varepsilon(s_1,t_1) \\
\varepsilon(s_2,t_1) \\
\vdots \\
\varepsilon(s_n,t_1) \\
\varepsilon(s_1,t_2) \\
\vdots \\
\varepsilon(s_n,t_T) \end{bmatrix} \in N \left( 0, \text{diag} \left[ \tau_\varepsilon(s_1), \tau_\varepsilon(s_2), \ldots, \tau_\varepsilon(s_n) \right] \right)$$

Collecting the $\tau_\varepsilon$ values into a $n \times 1$ vector we have a precision matrix for $\varepsilon$ with diagonal element given by

$$\text{diag} \left( Q_\varepsilon \right) = \begin{bmatrix} \tau_\varepsilon(s_1) \\
\tau_\varepsilon(s_2) \\
\vdots \\
\tau_\varepsilon(s_n) \end{bmatrix} = 1_{1 \times 1} \otimes \tau_\varepsilon$$

Space-Time Data — Estimation

To estimate the model you will be using Gibbs sampling

Write down the full posteriors for

- $p(\beta | y, \tau_\varepsilon, \tau_Q)$ — Reconstruction in a GMRF
- $p(\tau_Q | y, \beta, \tau_\varepsilon)$ — Gamma
- $p(\tau_\varepsilon | y, \beta, \tau_Q)$ — Gamma

Computations for $p(\tau_Q | y, \beta, \tau_\varepsilon)$ are very similar to those given for the corrupt pixels.
Matlab: Sparse matrices

- For sparse identity matrix use `speye` (not `eye`)
- For diagonal matrices use `spdiags` (not `diag`)
- Kronecker of sparse matrices is still sparse.
- When reordering $Q$ for e.g. simulation:
  
  ```matlab
  > %compute reorder
  > p = amd(Q);
  > %permute observation matrix (for mean)
  > A_p = A(:,p);
  > %permute and compute choleskey
  > R = chol(Q(p,p));
  > %sample or compute mean
  > X = R \ randn(size(R,1),1);
  > EX = R \ (R' \ (A_p'*Y));
  > %recover original order
  > X(p) = X; EX(p) = EX;
  ```