The Ising and Potts models for image restoration

Søren Højsgaard

Department of Mathematical Sciences

Aalborg University, Denmark

May 20, 2014

Contents

1 Image segmentation

Helle Thorning–Schmidt; Danish Prime Minister 2011–

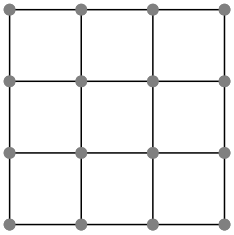
```
> library(pixmap)
> img <- read.pnm("./img/thorningS.ppm")
> plot(img)
```

A noisy grey scale image.

We want to de–speckle the image and segment it into contiguous regions.

Working model:

$$\text{observed image} = \text{true image} + \text{noise}$$



We want to estimate the true image from the observed image. Can be done in many ways.

We represent an image by a regular grid. Sites (pixels) are denoted by s or by (i, j). The observed pixel intensity at site s is denoted z_s (or by $z(i, j)$) and the true label (a colour) is denoted by x_s (or by $x(i, j)$).

The observed image is denoted $z = (z_s)$; the grid of labels (colours) is denoted $x = (x_s)$.

Neighbouring sites are likely to have the same labels.

The Bayesian paradigm is – in principle – well suited for this:

- We need a model for how the observed image is related to the true image; in statistical terms we need a likelihood
$$L(x) = p(z|x)$$
- We need a prior distribution over the true images; a distribution which for each image x will assign a probility to how probable this image is, namely
$$\pi(x)$$

With these two components in place the posterior becomes,

$$\pi^*(x) = p(x|z) = \frac{p(z|x)\pi(x)}{\sum_{x'} p(z|x')\pi(x')} = \frac{k(x)}{c}$$

and we can then sample from the posterior using Metropolis–Hastings type algorithms...

2 The Potts model

Suppose the label (colour) x_s at each site s can be one of the values $1, 2, 3, \ldots, L$.

For later use let $I(a, b)$ be an indicator function which is 1 if $a = b$ and 0 otherwise.

As a model for x we take the following variant of the Potts model:

$$\pi(x) = \frac{1}{c(\beta)} \exp\left(\beta \sum_{s,s'} I(x_s, x_{s'})\right)$$

where $\beta > 0$ is fixed parameter, $c = c(\beta)$ is the normalizing constant and the sum $\sum_{s,s'} \ldots$ is over all neighbouring sites s and s'.

Notice: A pair (s, s') of neighbouring sites with identical labels contributes to $p()$ by the amount $\exp(\beta)$. If the labels are different, the contribution is $\exp(0) = 1$.

Consider again

$$\pi(x) = \frac{1}{c(\beta)} \exp\left(\beta \sum_{s,s'} I(x_s, x_{s'})\right)$$

There is a detail here: Suppose each of the N pixels can have one of L different labels. The normalizing constant is then

$$c(\beta) = \sum_{x} \exp\left(\beta \sum_{s,s'} I(x_s, x_{s'})\right)$$

where the sum is over all L^N possible configurations of x.

Example: For a black and white image ($L = 2$) with 10×10 pixels (hardly enough to produce a smiley) the sum is over $2^{100} \approx 1.3e^{30}$ terms.

Hence the normalizing constant is infeasible to compute in practice. The prior is therefore known only up to a constant of proportionality.

2.1 Conditional independence

Recall that x and y are conditionally independent given z (written $x \perp\!\!\!\perp y \mid z$) if

$$p(x, y|z) = p(x|z)p(y|z)$$

or eqivalently if $p(x|y, z)$ does not depend on y, i.e.

$$p(x|y, z) = p(x|z)$$

or eqivalently if the joint density factorizes as

$$p(x, y, z) = g(x, z)h(y, z)$$

where $g()$ and $h()$ are non–negative functions (but they are in general not probability densities). This latter condition is called the factorization criterion.

In the latter case, it is "easy" to obtain $p(x|z)$ (which is the same as $p(x|y,z)$):

$$p(x|z) = p(x|y,z) = \frac{g(x,z)h(y,z)}{\int g(x,z)h(y,z)dx} = \frac{g(x,z)}{\int g(x,z)dx}$$

For a given site s we let $-s$ denote all sites except s. We also let $ne(s)$ denote those sites that are nearest neighbours of s. Lastly we let $r(s)$ denote "the rest", i.e. all other sites than s and $ne(s)$.

Now consider a particular site $\tilde{s}$. In

$$\pi(x) \propto \exp\big(\beta \sum_{s,s'} I(x_s, x_{s'})\big)$$

we group terms $I(x_s, x_{s'})$ according to whether they contain $x_{\tilde{s}}$ or not and obtain

$$\pi(x) \propto g(x_{\tilde{s}}, x_{ne(\tilde{s})})h(x_{ne(\tilde{s})}, x_{r(\tilde{s})})$$

where

$$g(x_{\tilde{s}}, x_{ne(\tilde{s})}) = \exp\big(\beta \sum_{s':\tilde{s}\sim s'} I(x_{\tilde{s}}, x_{s'})\big)$$

Hence

$$p(x_{\tilde{s}} \,|\, x_{ne(\tilde{s})}) = \frac{\exp\big(\beta \sum_{s':\tilde{s}\sim s'} I(x_{\tilde{s}}, x_{s'})\big)}{\sum_k \exp\big(\beta \sum_{s':\tilde{s}\sim s'} I(k, x_{s'})\big)}$$

So $\tilde{s}$ depends on the rest of the sites only through its neighbours: Given the label of the neighbours, the label of $\tilde{s}$ is independent of the label of all other sites.

Nice local property which will be used later on.

2.2 The observed image

Observed image: A grid $z = (z_s) = (z_{ij})$ of pixel intensities.

We can specify a joint probability model for unobserved labels x and the observed image z as

$$p(z,x) = p(z|x)p(x)$$

We have already specified $p(x)$ as the "Ising density", so we shall here specify $p(z|x)$. One way is by making additional conditional independence assumptions:

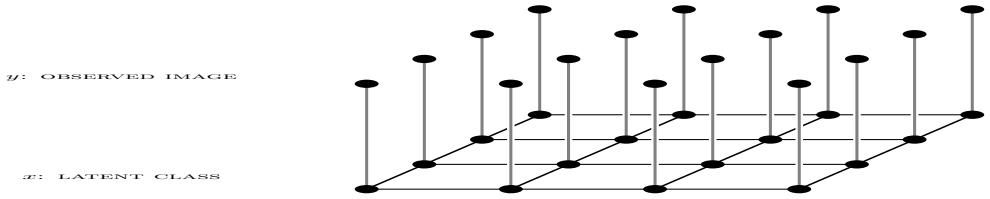
1. Assumption: The observations are conditionally independent given x:

$$p(z|x) = \prod_s p(z_s|x)$$

2. Assumption: Observation z_s at pixel s depends on x only through x_s (additional conditional independence restriction)

$$p(z_s|x) = p(z_s|x_s)$$

Notice: One possible choice for $p(z_s|x_s)$ is $p(z_s|x_s = j) \sim N(\mu_j, \sigma^2)$.



The structural form of $\pi^*(x) = k(x)/c$ is therefore

$$\log k(x) = \sum_s \log p(z_s|x_s) + \beta \sum_{s,s'} I(x_s, x_{s'})$$

3 A naive sampling scheme

We shall implement a Metropolis–Hastings sampler for this model to produce a sequence $x^1, x^2, \ldots$ of "true images". Given the current value x^t. Two naive (and very inefficient) suggestions:

- Update all of x: Generate a new random image x^P as proposal. Hence the proposal distribution $h(x)$ is uniform and does not depend on x^t.
- Update a single component of x: Generate x^P as follows: Set $x^P = x^t$. Pick a random site s. Propose a random label x_s for that site and plug that label into x^P leaving everything else unchanged. In this case $h(x|x^t) = h(x^t|x)$ is symmetric.

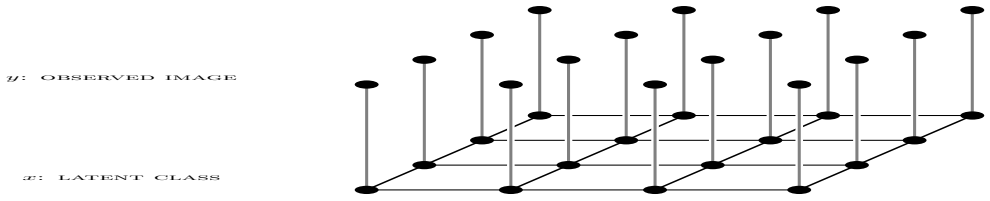
In either case, all we need is therefore to compute

$$\alpha = \min\left(1, \frac{k(x^P)}{k(x^t)}\right)$$

and accept x^P with probability α.

3.1 Single component updates

We can make a more efficient single component update by exploiting the conditional independence structure in the model.



Consider a specific site $\tilde{s}$. In

$$\pi(x) \propto \big(\prod_s p(z_s|x_s)\big) \exp\big(\beta \sum_{s,s'} I(x_s, x_{s'})\big)$$

we group terms according to whether they contain $x_{\tilde{s}}$ or not and obtain

$$\pi(x) \propto g(x_{\tilde{s}}, x_{ne(\tilde{s})}) h(x_{ne(\tilde{s})}, x_{r(\tilde{s})})$$

where

$$g(x_{\tilde{s}}, x_{ne(\tilde{s})}) = p(z_{\tilde{s}}|x_{\tilde{s}}) \exp\big(\beta \sum_{s':s'\sim\tilde{s}} I(x_{\tilde{s}}, x_{s'})\big)$$

Reasoning as before we find that

$$\begin{aligned} p(x_s = j|x_{-s}, z) &= p(x_s = j|x_{ne(s)}, z_s) \\ &= \frac{p(z_s|x_s = j) \exp\big(\beta \sum_{s':s'\sim s} I(j, x_{s'})\big)}{\sum_k p(z_s|x_s = k) \exp\big(\beta \sum_{s':s'\sim s} I(k, x_{s'})\big)} \\ &\propto p(z_s|x_s = j) \exp\big(\beta \sum_{s':s'\sim s} I(j, x_{s'}) \end{aligned}$$

where $p(z_s|x_s = j)$ is a normal density $N(\mu_j, \sigma^2)$.

Hence the conditional distribution is a discrete distribution over the values $j = 1, 2, \ldots, L$, and this distribution is easy to simulate from. Hence the Gibbs sampler is easy to implement.

Now it is straight forward to propose a more efficient sampling scheme. Think of it as an outer loop and an inner loop.

At the entry of a the tth iteration of the outer loop, let x^{t-1} be the current value.

Set $x^t = x^{t-1}$.

- Pick a site $\tilde{s}$ with label $x_{\tilde{s}}$ in x^t.
- Propose a new label $x^P_{\tilde{s}}$ from a proposal distribution $h(\cdot|x^t)$.
- Calculate the acceptance probability

$$\alpha = \min\big\{1, \frac{g(x^P_{\tilde{s}}, x_{ne(\tilde{s})})}{g(x_{\tilde{s}}, x_{ne(\tilde{s})})} \frac{h(x_{\tilde{s}}|x^t_{-\tilde{s}}, x^P_{\tilde{s}})}{h(x^P_{\tilde{s}}|x^t_{-\tilde{s}}, x_{\tilde{s}})}\big\}$$

- Accept $x_{\tilde{s}}^P$ with probability α. If accepted, stick $x_{\tilde{s}}^P$ into x^t at site $\tilde{s}$
- Move on to the next site

When all sites have been visited we are done with the inner loop and we have and updated x^t. Go to the next step in the outer loop.

This way we obtain a sequence of images $x^1, x^2, \ldots$ from the posterior $\pi^*(x)$.

3.2 Helle Thorning

Helle Thorning–Schmidt; Danish Prime Minister 2011–

```
> library(pixmap)
> img <- read.pnm("./img/thorningS.ppm")
> plot(img)
```

A noisy grey scale image.

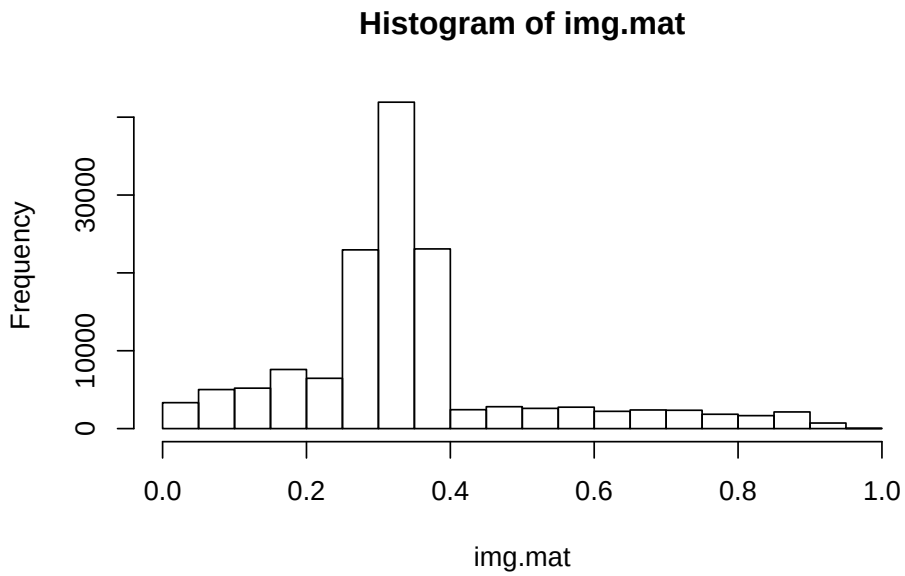
```
> img
```

```
Pixmap image
  Type          : pixmapGrey
  Size          : 300x465
  Resolution    : 1x1
  Bounding box  : 0 0 465 300
```

```
> img.mat <- img@grey
> dim(img.mat)
```

```
[1] 300 465
```

```
> hist(img.mat)
```



Pixel values are between 0 and 1.

We assume that the true underlying image is a grid of pixels labelled $1, 2, \ldots, L$ and that

$$z_s \,|\, x_s = j \sim N(\mu_j, \sigma^2)$$

As μ_js and σ we take

```
> ( sdval   <- sd(as.numeric(img.mat)) )
[1] 0.1703555
> ( muvec   <- seq(0.1,0.9,.1) )
[1] 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9
```

3.3 Starting values: ML classification

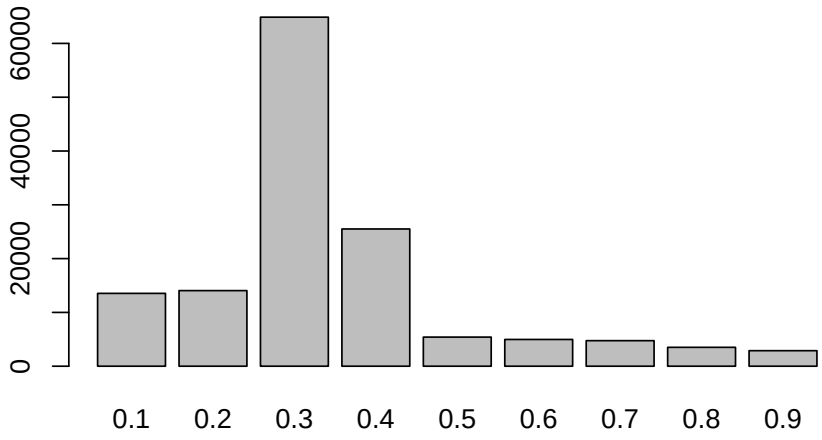
In principle we can start the sampler anywhere (i.e. from any initial grid of labels); in practice we start with the result of running a maximum likelihood classification.

```
> mlclass <- function(img.mat, mean, sd){ # Homegrown function
+   cls <- img.mat
+   for (ii in 1:nrow(cls)){
+       for (jj in 1:ncol(cls)){
+           cls[ii, jj] <- which.max(dnorm(img.mat[ii, jj],
+                                          mean, sd, log=T))
+       }
+   }
+   fit   <- cls; fit[] <- mean[cls]
+   list(label=cls, fit=fit, img=pixmapGrey(fit))
```

```
+ }
> ## ML classification of each pixel
> mlc <- mlclass(img.mat, muvec, sdval)
> names(mlc)
[1] "label" "fit"   "img"
> plot(mlc$img)
```

```
> table(mlc$fit)

  0.1   0.2   0.3   0.4   0.5   0.6   0.7   0.8   0.9
13537 14053 64879 25507  5410  4970  4748  3508  2888
> barplot(table(mlc$fit))
```

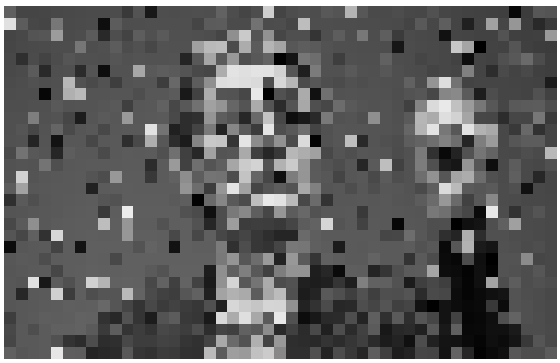


3.4 QUIZ

Notice: This will require some (home)work!

- Implement the single component sampler described above. As proposal distribution just take a uniform on $1, 2, \ldots L$.
- Hint: In R, `sample()` is your friend.
- Hint: If you know how to program in a compiled language such as C or C++ then it is suggested that you do so.
- How sensitive is the algorithm to starting values?
- Is there a scope for parallel computing here?
- Hint: When implementing the sampler you might find it convenient to first work with a smaller image:

```
> dim(img.mat)
[1] 300 465
> seq(1, 300, by=10)
 [1]   1  11  21  31  41  51  61  71  81  91 101 111 121 131 141 151 161 171 181
[20] 191 201 211 221 231 241 251 261 271 281 291
> img.mat2 <- img.mat[seq(1, 300, by=10), seq(1, 465, by=10)]
> pixmapGrey(img.mat2)
Pixmap image
  Type           : pixmapGrey
  Size           : 30x47
  Resolution     : 1x1
  Bounding box   : 0 0 47 30
> plot(pixmapGrey(img.mat2))
```



- Here is a result after running the sampler for x^{50} when starting the sampler with the result of a maximum likelihood classification.

```
> plot(img50$img)
```