Tina Memo No. 2015-008

Internal.

Linear Poisson Models; For the Rest of Us.

J. Gilmour.

Last updated
17 /15 / 2015

Imaging Science and Biomedical Engineering Division,
Medical School, University of Manchester,
Stopford Building, Oxford Road,
Manchester, M13 9PT.

.

Preface

*Our original publication of this work was in the form of a conventional paper (see Tina-Memo 2014-009), with the usual paragraphs of text used to motivate the ideas, formulate the problems and justify solutions. While this presentation is perhaps amenable to experienced statisticians and algorithm developers, it is not easily understandable to "***the rest of us***" (as Jamie put it).*

This document contains Jamie's condensed, key point based, description of Linear Poisson Models and their Application to image derived histogram data. The advantage of this presentation is that it is possible to see at a glance the various problems which had to be solved when developing this approach. The document can be read either as a key summary (by reading only the top level sections) or at any level of detail required to understand any of the computations (by reading subsections). As such it may provide a better way of understanding what was done than the original publication, particularly if you wish to try to re-code any of it. In this form it is probably also easier to focus on the key mathematical derivations, should you wish to re-derive any of them for yourself.

It should be noted that all approximations given are as used (and tested) in Paul Tar's thesis and this resulting publication. Jamie's notation differs from the original work in some places, but this may not be a bad thing if you are having difficulty with the previous versions. Another relatively condensed form of this work can also be found in our BMVA Annals publication (see Tina-Memo 2013-006).

N.Thacker

Quantitative Image Analysis with Linear Poisson Models (for the Rest of Us).

Jamie Gilmour (based on Tar et al. (2015) Advances in Space Research, **56**, 92-105)

Introduction

What is the technique "Linear Poisson Models" (LPM)? Suppose you have a dataset in the form of a histogram where each bin contains a number of counts. Your hypothesis is that the corresponding sample is a linear sum of various components. For each component you have a suite of histograms derived from similar analysis of representative examples. The LPM technique allows you to test your hypothesis that the sample is made up of a linear combination of the components you have identified. If the hypothesis survives, LPS allows you to determine how much of each component is present. The advantage of LPM over other methods is that it rigorously determines the uncertainties in the component histograms and in the sample histogram, and propagates them to make the hypothesis test rigorous and to determine the uncertainties in the quantities of components present in the sample.

This document fleshes out the derivations presented in our paper (referenced above). In this paper we apply the technique to image analysis. Some sections below relate to the method of converting an image into a histogram in a way that allows the LPM technique to be applied. Those wishing to apply LPM in other contexts may wish to skip over them and start at section 7, bearing in mind that from this point forward images are discussed in terms of the histograms derived from them.

What follows amplifies the explanation and derivations in the paper referenced above and is aimed at those who do not routinely work in this field.

Preamble

1. Our goal is to extract quantitative measurements from images in a way that allows scientific inferences to be made reliably. By this we mean that the derived quantities have errors corresponding to uncertainties in the input data. As an example, we tackle the problem of identifying what fractions of an imaged area are made up of various different terrains.
   - 1.1. Each terrain is defined with reference to a set of training images that are examples of that terrain.
   - 1.2. The target image is modelled as a combination of the various terrains.
   - 1.3. Our goal requires uncertainties to be estimated on derived quantities.
   - 1.4. We adopt the principle that the probability of an event can be estimated from the frequency with which it is observed in a large number of opportunities.
2. A successful method would:
   - 2.1. Tell us what fraction of the image was made up of each terrain.
   - 2.2. Tell us what the uncertainty was on each of these measurements.
     - 2.2.1. Uncertainty arises from both the training images and the image being analysed.
     - 2.2.2. It is necessary to find a quantitative process where variances can be calculated and combined to produce an error estimate for each derived quantity.
   - 2.3. Tell us whether the image was in fact consistent with being made up of the terrains we were looking for, or whether unexpected features also contribute.
     - 2.3.1. This requires a goodness-of-fit statistic that takes into account variances on quantities defined in the process so that it can be compared to the number of degrees of freedom in the model.
   - 2.4. Items 2.2 and 2.3 both require that variances be derived rigorously from data.
3. In section 4 we summarise the information in an image in a form amenable to statistical analysis. In sections 6 to 9 we characterise a terrain based on a series of training images and define a goodness of fit statistic. In section 11 we derive the uncertainties on the terrain parameters. In section 12 we show

how to find the best model of an image using a set of terrains. In section 13 we combine the errors arising from the input image and the terrain definitions to produce errors on the model parameters.

3.1. Each pixel in an image is treated as the centre of a local neighbourhood.

3.2. The pattern in that neighbourhood is characterised by a sampling method that produces a binary vector.

3.3. For our statistical treatment we will want to treat the instances of each vector present in an image as independent. However, the same underlying process may have caused contiguous pixels to have the same neighbourhood pattern – they are not independent. For this reason we group assemblies of such contiguous “identical” pixels into blobs. The blob vector includes information about blob size in addition to the neighbourhood pattern.

   3.3.1. Of course, a process may reproducibly introduce pairs of different vectors that appear together (and so on); we have not eliminated the problem entirely by constructing blobs in this way.

3.4. We summarise the information as a histogram that records the number of instances of each blob vector in the image.

3.5. Histograms extracted from training images are used to characterise a *terrain*. A *terrain* is made up of component *textures*. A training image contains examples of the component textures in different proportions.

3.6. We derive a set of texture histograms for each terrain from the exemplar training images.

3.7. We calculate uncertainties on these histograms.

3.8. We find the optimum combination of terrains that model the image.

3.9. We derive uncertainties on the proportions of each terrain in the image.

Summarising the Information in an Image as a Histogram.

4. We first need to summarise the information in an image in a way that allows us to develop such a method.
   4.1. The first step is summarise the pattern in the neighbourhood of each pixel.
      4.1.1. To do this we select a series of pairs of pixels in the neighbourhood of a reference pixel.
         4.1.1.1. In any analysis the positions of the pixel pairs are always the same relative to the reference pixel.
      4.1.2. In each pair, if the first pixel is brighter than the second we assign a 1, if darker we assign a 0. The neighbourhood can then be summarised as a binary vector e.g. (0,1,1,0,1,0,1,1). This is based on the BRIEF representation.
      4.1.3. In this way, the neighbourhood of each pixel in the image can be assigned to one of 2^8 = 256 different patterns.
      4.1.4. The image could then be summarised as a histogram with 256 classes corresponding to the 256 possible neighbourhood patterns that counted the number of occurrences of each in the image.
   4.2. At this point we must remember that we want to propagate uncertainties from this initial measurement.
      4.2.1. Because we have counted the number of instances of each neighbourhood pattern in the image, we can estimate the uncertainty in this number using Poisson statistics. If a random event is observed *N* times over a domain, the uncertainty on the measurement of events/domain is $\sqrt{N}$. The variance is *N*.
         4.2.1.1. In what follows we will usually discuss variances.
      4.2.2. Using Poisson errors from our neighbourhood pattern counts is only valid if instances of a pattern are uncorrelated. We are assuming that the separate observations of a pattern are independent of one another.
      4.2.3. In reality, it is common for a process to result in the same pattern *p* being found in the neighbourhood of several contiguous pixels, so the assumption that they are independent is incorrect.
         4.2.3.1. This is because the two pixels are part of the same feature, so only result from one event.
   4.3. To overcome this we extend the definition of our histogram elements.
      4.3.1. We add 4 elements to our vector that encode the size of a contiguous area of identical neighbourhood patterns.
      4.3.2. If the pattern is isolated these 4 elements are (0,0,0,0). If it is two pixels, (0,0,0,1). If three or four pixels (0,0,1,0), and so on. (0,1,1,1) would thus encode any contiguous area of the same pixel that appeared between 64 and 128 times.
   4.4. We now have a histogram that summarises the information in the image. It has 4096 elements, each counting the number of times a *blob* of identical neighbourhood patterns of a particular size range appears in the image. This has been formulated in such a way as to allow the uncertainty on each element to be assessed rigorously. For instance, if we have 100 blobs of a certain type in an image, we associate a Poisson variance of 100 (no. of counts = 100 ± 10).
   4.5. **Terminology:** We use *x* to denote a particular 12 element vector that represent a blob. A histogram that tells us the frequency of each *x* in an image *z* is $H_{x(z)}$.

Determining Input Terrains from Exemplar Images.

5. To divide up an image into areas covered by various terrains we need to define a terrain.
   - 5.1. **Terminology:** We propose that a *terrain* is made up of characteristic *textures*. In any example of this terrain, these characteristic textures will be the same, but they will be present in different proportions.
   - 5.2. **Terminology:** When we say a texture is "the same", we mean that different examples of this texture will produce the same shape histogram (within error) when encoded using the method of section 5.
6. We start with a dataset of training images of this terrain.
   - 6.1. We use the method of section 5 to produce a histogram for each of these images.
   - 6.2. From 5.2, it follows that each one of these terrain-histograms corresponds to a sum of the texture-histograms that correspond to the terrain's textures. The *terrain-histograms* vary from one training image to another because the *texture-histograms* are present in different proportions in different training images.
   - 6.3. The challenge is to determine what these texture-histograms are. To do this we use the set of terrain-histograms derived from the set of training (exemplar) images of the same terrain.
   - 6.4. Each training-histogram is made up of (unknown) proportions of the same (unknown) texture-histograms. Solving this problem is a form of Independent Component Analysis. We use an expectation-maximization technique, which is iterative.
     - 6.4.1. We guess a set of texture-histograms.
     - 6.4.2. We produce an optimised model of the training image set by varying the proportions of the texture histograms.
     - 6.4.3. From the optimised model we estimate the fraction of each texture contributing to each bin of the image histograms.
     - 6.4.4. We apply these fractions to the actual image histograms to create improved texture-histograms.
     - 6.4.5. We produce a new model based on the new texture-histograms and repeat the process.
     - 6.4.6. Eventually iterating the model leads to a consistent set of texture histograms and contributions to the training image.
7. Suppose we have *N* training images for a terrain. We have produced a histogram $H_{x(r)}$ from each training image *r*. **The "Linear Poisson Models" method is described below in terms of image analysis, but can be applied to any circumstances where a "sample" histogram is hypothesised to be a linear combination of "components", where there are multiple examples of each component, and where the errors on the histogram bins values obey Poisson statistics.**
   - 7.1. We model this terrain as made up of *n* textures. The k^th texture generates blob *x* with a probability mass function *P(x|k)*. (In texture "*k*", *"P(x|k)"* of the blobs will be of type "*x*".) There is $Q_{k(r)}$ of this texture in each image *r*. In other words…
   - 7.2. $$H_{x(r)} \text{ is modelled by } \sum_{k=1}^{n} P(x|k)Q_{k(r)} = M_{x(r)}$$
     - 7.2.1. For instance, the amount of blob *x* in image *r* can be found by adding together the amount of texture 1 times the probability that *x* appears in texture 1, the amount of texture 2 times the probability that *x* appears in texture 2, the amount of texture 3 times…
   - 7.3. Our goal is to estimate the *P(x|k)* (the frequency with which blob x occurs in texture k). In other words we need to find *P(x|k)* and the $Q_{k(r)}$ that sufficiently model the complete set of training images. We use the same *P(x|k)* for each image, the variation between training images is captured by $Q_{k(r)}$.

Optimization of Terrain Texture Descriptions.

8. What do we mean by "sufficiently model"? In this case, we need a goodness of fit parameter, χ^2, that tells us whether our set of *n* textures adequately accounts the terrain. We have a model that uses *n* textures, each of which has a $P(x|k)$, to model the variation across all the $H_{x(r)}$ (which includes one histogram for each training image *r*). They have to account for the variation in $H_{x(r)}$ across *N* training images. Each image has a set of weights $Q_{k(r)}$. We need to find the smallest number of textures (the smallest *n*) that can be used with various weights to bring χ^2 to an acceptable level (a value close to the number of degrees of freedom of the model. (χ^2 is defined in section 9.)

8.1. To do this we need a method for finding the best $P(x|k)$ and $Q_{k(r)}$ for a given $H_{x(r)}$ and *n.* (For the model with *n* textures, what is the best combination of textures and weights to model the suite of training images?) This is done iteratively.

8.2. Suppose we have a set of $P(x|k)$ for a terrain and a set of $Q_{k(r)}$ for a training image $H_{x(r)}$. As in 7.2 we can construct a histogram $M_{x(r)}$ such that

8.3. $$M_{x(r)} = \sum_{k=1}^{n} P(x|k)Q_{k(r)}$$

8.4. $M_{x(r)}$ is a model of the training image $H_{x(r)}$ based on the $P(x|k)$.

8.5. The fraction of the contribution from a single texture *k* to any element *x* of *M* is then

8.6. $$F(x|k)_r = \frac{P(x|k)Q_{k(r)}}{M_{x(r)}} = P(k|x)$$

8.6.1. In other words, the probability that texture *k* generated blob *x* is estimated as the fraction of blob x in the current model that was produced by *k*.

8.6.2. We can use this with the measured $H_{x(r)}$ to produce a new estimate of $P(x|k)_r$ based on the r^{th} training image. First we define

8.6.3. $$R(x|k)_r = F(x|k)_r H_{x(r)} \qquad (= P(x|k)_r Q_{k(r)})$$

8.6.3.1. In other words… In our model a certain fraction of blob *x* is contributed from each texture *k.* We divide up the amount of blob *x* in the actual image histogram among the various textures in the same proportions giving $R(x|k)$, which is the number of instances of blob *x* attributed to texture k_i in training image histogram H_x. This in effect allows us to calculate a new $P(x|k)_r$ using the current set of Q_k.

8.6.4. The $R(x|k)$ for the various training images are then combined to provide new estimates of $P(x|k)$.

8.6.5. $$P(x|k)_{\text{new}} = \frac{\sum_{r=1}^{N} R(x|k)_r}{\sum_{r=1}^{N} Q_{k(r)}}$$

8.6.5.1. For each blob, the probability that it is produced by a particular texture is calculated as the weighted average of the frequencies (see **Error! Reference source not found.**) for that blob in that texture deduced from the training images. Loosely speaking, this is the total amount of *x* from *k* divided by the total amount of *k* called for in each image.

8.7. Having updated the $P(x|k)$ we need to update the $Q_{k(r)}$.

8.7.1. A blob may be part of several terrains. At any iteration we have the $P(x|k)$ and the $Q_{k(r)}$. The probability that a particular example of a blob *x* came from texture *k is*

8.7.2. $$P(k|x) = \frac{P(x|k)Q_{k(r)}}{\sum_{k=1}^{n} P(x|k)Q_{k(r)}}$$

8.7.2.1. Suppose we have three textures. We expect that the number that came from texture 1 is $Q_1P(x|1)$, the number from texture 2 is $Q_2P(x|2)$, the number from texture 3 is $Q_3P(x|3)$. The fraction of the expected total that came from any one texture (8.6.2) is our current estimate of the probability that the texture produced any particular example of that blob in the image.

8.7.3. and the new weight is

8.7.4. $$Q_{k(r)} = \sum_x P(k|x)H_x$$

8.7.4.1. We weight the estimate of the probability by the number of examples of that blob in the image. We sum these weights over the all the blobs in the image.

8.7.5. One set of $Q_{k(r)}$ have been used in 8.7.2 to produce a new set in 8.7.4.

8.7.6. Note that the $Q_{k(r)}$ are not normalised.

8.8. We alternate steps 8.6 (updating *P* given a set of *Q*) and 8.7 (updating *Q* given a new set of *P*) until the $P(x|k)$ and $Q_{k(r)}$ no longer change. At this point we have found the optimum set of $P(x|k)$ and $Q_{k(r)}$ for modelling the training images with *n* textures.

8.9. Once optimised and with an acceptable χ^2 per degree of freedom we carry forward a histogram representing the texture *k*.

8.10. $$H_{x|k} = \sum_r R_r(x|k)$$

Goodness-of-Fit Parameter

9. To complete this we need χ^2 per degree of freedom to evaluate whether we have successfully modelled the set of training images with our *P(x|k).*

9.1. How many degrees of freedom are there? In effect we are proposing textures of a fixed form, and attempting to model the suite of histograms from training images. There are *n* textures. There are *N* training images. Each histogram has *m* entries each of which corresponds to a vector *x*. The Model population of the bin corresponding to vector *x* is $M_{x(r)}$. The actual population is $H_{x(r)}$.

9.1.1. The distribution of counts in a bin is Poisson. We need to take the square root of this to make the distribution close to a Gaussian with width such that $\sigma^2 = ¼$. This is a standard result.

9.1.2. The contribution to the goodness of fit from the vector *x* in training image histogram *r* is then…

9.1.3. $$4(\sqrt{M_{x(r)}} - \sqrt{H_{x(r)}})^2$$

9.2. Each image histogram has *m* entries and is fitted by varying the quantities of the *n* textures. There are *N* image histograms each thus contributing *(m-n)* degrees of freedom. So χ^2 per degree of freedom is

9.3. $$\chi_D^2 = \frac{4}{N(m-n)} \sum_r \left(\sum_x (\sqrt{M_{x(r)}} - \sqrt{H_{x(r)}})^2\right)$$

Summary and Review.

10. In section 4 we presented a way of summarising the information in an image in a numerical form amenable to statistical analysis. In sections 6 to 9 we developed a quantitative approach to characterising a terrain based on the numerical summaries of a series of training images.
    - 10.1. Each pixel in an image is treated as the centre of a local neighbourhood.
    - 10.2. The pattern in that neighbourhood is characterised by a sampling method that produces a binary vector.
    - 10.3. The same underlying process may have caused contiguous pixels to have the same neighbourhood pattern. For our statistical treatment we will want to treat the instances of each vector present in an image as independent. For this reason we group assemblies of such contiguous "identical" pixels into blobs. The blob vector includes information about blob size in addition to the neighbourhood pattern.
    - 10.4. We summarise the information as a histogram that records the number of instances of each blob.
    - 10.5. Histograms extracted from training images are used to characterise a terrain. A terrain is made up of component textures. A training image contains examples of the component textures in different proportions.
        - 10.5.1. Working with the training image histograms, we have shown how the set of component texture histograms can be extracted from the suite of training images.

Uncertainty on Texture Histograms

11. We now need to consider the sources of error that contribute to the derived $H_{x|k}$ (8.10).
    11.1. We start from equations 8.6 and 8.6.3
    11.2. $F(x|k)_r = \frac{P(x|k)Q_{k(r)}}{M_{x(r)}} = P(k|x)$ (8.6) $R(x|k)_r = P(k|x)_r H_{x(r)}$ (8.6.3)
    11.3. The variance in $P(x|k)_r$ comes from independent variances in $P(k|x)$ and $H_{x(r)}$.
        11.3.1. The fraction $F(x|k)_r$ is established in 8.6. It is the number of instances of a particular *x* in a **model** of image *r* that came from texture *k,* expressed as a fraction of the total number of instances of that *x* in the model. It is assumed that this is the same as the fraction of the total number of instances of *x* in the **image** that come from texture *k* and this is how it is used in 8.6.3 as $P(k|x)$.
            11.3.1.1. There are $H_{x(r)}$ instances of blob *x* in image *r*. Of these we expect that $P(k|x)H_{x(r)}$ come from texture *k.*
            11.3.1.2. If we looked at multiple equivalent images we would not expect to see exactly the same number of instances of *x* in each one – there would be a distribution corresponding to the probability that, in random process with a given frequency f, we would observe exactly *m* instances out of a total of *n* trials. This is a binomial distribution and the variance is $nf(1-f)$.
        11.3.2. Our working assumption is that individual instances of blob *x* are independent, so the number *n* observed in any image follows a Poisson distribution and the variance is *n.*
        11.3.3. In our case $n = H_{x(r)}$ and $f = P(k|x)$. The expected number of observed instances is $H_{x(r)}P(k|x)$. The variance on the number of instances is $H_{x(r)}P(k_i|x)(1-P(k_i|x))$. The standard deviation is the square root of this. Divide through by $H_{x(r)}$ to get the fraction and the s.d. of the fraction, and square the latter to get the variance on the fraction as $P(k_i|x)(1-P(k_i|x))/H_{x(r)}$.
    11.4. Combining variances in the usual way i.e. by taking partial differentials of 8.6.3, squaring them, multiplying them by the variances and adding, the variance on $R(x|k)$ is:
    11.5. $$\frac{H_{x(r)}^2 P(k|x)\big(1-P(k|x)\big)}{H_{x(r)}} + P(k|x)^2\, H_{x(r)} = H_{x(r)}P(k|x) = R(x|k)$$
    11.6. So the $R(x|k)$ can be treated as having a Poisson distribution in this respect.
    11.7. This allows us to calculate uncertainties on the $H_{x|k}$ derived from training images.

Modelling an Image Histogram as Composed of Terrain Texture Histograms

12. In sections 8 and 9 we have presented a means of extracting texture histograms from a set of training images that are exemplars of that texture. The problem of analysing an image is related to this and a similar technique can be employed.

    12.1. Rather than seeking to find the set of textures k that make up the image, we attempt to find the best model for the image that uses the already defined textures. We employ the χ^2 per degree of freedom of this best fit model as a measure of whether it successfully modelled the image under analysis. The Q_k that correspond to this best fit allow the areas covered by the various textures to be determined. The association of each texture with a parent terrain then allows the area covered by each terrain to be determined.

    12.2. We repeat the steps in (8.7) to optimise the values of Q_k.

Uncertainty on Derived Model Parameters

13. Once we have calculated a solution to any problem, we want an estimate of the uncertainty associated with its parameters. Specifically, we want to estimate how far from the "true" solution our calculated solution is. In this case the separation arises from the input uncertainties in the training images and in the image being analysed.
    13.1. Where a derived quantity is calculated directly from observed values by a formula, observational errors can (at least in principle) be propagated to the derived quantities. The principle is used in 14.1 below.
    13.2. Where necessary, quantities involved in determining the variances are calculated at the values corresponding to the best solution.
    13.3. For instance, if $y = e^{-kx}$ $\quad \Delta y^2 = \left(\frac{\partial y}{\partial x}\Delta x\right)^2 = (ky\Delta x)^2$ and we would evaluate *y* at the best fit" corresponding to the observation *x* in calculating *Δy*.
    13.4. We have used an iterative process. We can use the above approach to calculate the uncertainty arising from one step, but this will underestimate the uncertainty arising from the complete iterative process of finding a solution.
    13.5. The "one step" error will be "amplified" – the error from the first step will be propagated to the second, and so on.
    13.6. We now present a means to estimate the one step error and the amplification factor.
    13.7. In each case, all quantities used are evaluated at the "best fit" solution determined as above.14.1
14. We seek a covariance matrix C_{EMStep} for **a single step** that records the variances of the Q_k and their interdependence. These arise from the uncertainties associated with the input image histogram H_x and with the terrain histograms determined from exemplar images $H_{x|k}$. The elements of the matrix are thus…
    14.1.
$$C_{ij} = \sum_x \left[\left(\frac{\partial Q_i}{\partial H_x}\right)\left(\frac{\partial Q_j}{\partial H_x}\right)\sigma_{H_x}^2 + \sum_k \left(\frac{\partial Q_i}{\partial H_{x|k}}\right)\left(\frac{\partial Qj}{\partial H_{x|k}}\right)\sigma_{H_{x|k}}^2\right]$$
$$= \sum_x \left[\left(\frac{\partial Q_i}{\partial H_x}\right)\left(\frac{\partial Qj}{\partial H_x}\right)\sigma_{H_x}^2\right] + \sum_x \left[\sum_k \left(\frac{\partial Q_i}{\partial H_{x|k}}\right)\left(\frac{\partial Qj}{\partial H_{x|k}}\right)\sigma_{H_{x|k}}^2\right]$$
$$= C_{ij(data)} \qquad + \qquad C_{ij(model)}$$
    Where the variables in the last line correspond to the sums over *x* directly above them.
    14.2. Since the update to $Q_k = P(k|x)H_x$, $\quad C_{ij(data)} = \sum_x P(i|x)P(j|x)H_x$.
    14.3. An expression for $C_{ijmodel}$ is harder to obtain.
        14.3.1. We first consider a single element *x* and a single texture *k*. We label $\bar{x}$ all the vectors apart from *x* and $\bar{k}$ all the textures apart from k.
        14.3.2. We start from $Q_k = P(k|x)H_x + P(k|\bar{x})H_x$. This is an expansion of 8.7.4 as applied to the problem of analysing one image in terms of a set of known textures.
        14.3.3. *P(k|x)* is the fraction of the number of instances of *x* that originate from texture *k* in the current model. If the frequency with which *x* is produced by the texture *k* is $F_m(x|k)$ then
        14.3.4.
$$P(k|x) = \frac{F_m(x|k)Q_k}{F_m(x|k)Q_k + F_m(x|\bar{k})Q_{\bar{k}}}$$
        14.3.5. This is the number of instances of *x* caused by *k* divided by the total number of instances of *x* (i.e. those from k and those from the $\bar{k}$. The $F_m(x|k)$ are calculated from the histograms that describe the texture (recall that we showed in 11 that the bin entries have Poisson errors).
        14.3.6.
$$\text{and} \quad F_m(x|k) = \frac{H_{x|k}}{H_{x|k} + H_{\bar{x}|k}} \qquad \text{and} \qquad F_m(x|\bar{k}) = \frac{H_{x|\bar{k}}}{H_{x|\bar{k}} + H_{\bar{x}|\bar{k}}}$$

14.3.7. Which, with 14.3.4, allows the derivatives in $C_{ijmodel}$ to be calculated from 14.3.2 with some standard long winded differentiation.

15. We now consider how errors are propagated through the iterative process.
    15.1. The entries in C_{EMStep} can be thought of as $C_{\mathrm{EMStep}ij} = \frac{1}{n}\sum_n\left(\Delta_{\mathrm{EMStep}\ i}\Delta_{\mathrm{EMStep}\ j}\right)_n$
        15.1.1. $\Delta_{\mathrm{EMStep}\ d}$ is the deviation from the expected value of Q_d in one EM step in the n^{th} of an (imaginary) series of nominally identical analyses. The deviations arise from uncertainties (ie statistical variation across the *n* repeats) associated with the input image histogram H_x and with the terrain histograms determined from exemplar images $H_{x|k}$.
    15.2. We seek a covariance matrix for the complete EM process, which would have the similar form $C_{ij} = \frac{1}{n}\sum_n\left(\Delta_i\Delta_j\right)_n$
        15.2.1. Δ_d is the deviation at the end of the EM process from the true value of Q_d arising from the uncertainties associated with the H_x and the $H_{x|k}$ each of the *n* repeats from the "true".
    15.3. Suppose after one step in one of the *n* analyses each of the *Q* is **Δ** from the correct value (i.e. the mean of all the *n* analyses owing to the uncertainties in H_x and $H_{x|k}$ (i.e. the difference between them and the means of these histograms across the *n* analyses).
        15.3.1. **Δ** is a vector recording the Δ_i for each Q_i. Note that $C \propto \sum_n(\Delta^T\Delta)_n$ from 15.2.
    15.4. At the next iteration an additional deviation from the correct value is introduced, where
    15.5. $\Delta_{+1} \approx (\Delta)\nabla\mathrm{Q}$ where the Jacobian $\nabla\mathrm{Q}_{ij} = \frac{\partial\Delta_{i,+1}}{\partial\Delta_j}$ (note the order) because $Q = Q_{\mathrm{true}} + \Delta$ and so $\frac{\partial Q}{\partial x} = \frac{\partial\Delta}{\partial x}$.
    15.6. Combining 8.7.2 and 8.7.4, adopting the terminology from section 14, and applying to the case of analysing an image we have:
    15.7. $$Q_{k(+1)} = \sum_x \frac{P(x|k)Q_k}{P(x|k)Q_k + P(x|\bar{k})Q_{\bar{k}}} H_x$$
    15.8. For on axis cases, differentiating leads to…
    15.9. $$\nabla Q_{ii} = \sum_x P(x|i) - \frac{P(x|i)^2Q_i}{H_x} = \sum_x P(x|i) - P(x|i)P(i|x)$$
    15.10. While for off axis cases it leads to
    15.11. $$\nabla Q_{ij} = \sum_x -P(x|j)P(i|x)$$
    15.12. On the assumption that the derivatives are approximately equal at every step, as the number of steps increases arbitrarily the total error approaches.
    15.13. $$\Delta \approx \Delta_0[\mathbf{I} - \nabla\mathbf{Q}]^{-1} = \Delta_0 S$$
        15.13.1. $\Delta = \Delta_0 + \Delta_0\nabla\mathrm{Q} + \Delta_0\nabla\mathrm{Q}^2 + \cdots$ so $E = \Delta_0(\mathbf{I} + \nabla\mathrm{Q} + \nabla\mathrm{Q}^2 + \cdots)$
        15.13.2. Let $SUM = \mathbf{I} + \nabla\mathrm{Q} + \nabla\mathrm{Q}^2 + \cdots$ so $SUM = \mathbf{I} + S\nabla\mathrm{Q}$ so $SUM(\mathbf{I} - \nabla\mathrm{Q}) = \mathbf{I}$
        15.13.3. so $SUM = (\mathbf{I} - \nabla\mathrm{Q})^{-1}$
    15.14. This scaling factor tells us how each of the $\Delta_{\mathrm{EMStep}\ d}$ in 15.1 can be related to the corresponding Δ_d in 15.2. In order to complete the scaling we produce the final covariance matrix through 15.3.1 (i.e. summarising the variation across the ensemble of *n* hypothetical repeats).
        15.14.1. $C \propto \sum_n(\Delta^T\Delta)_n = \sum_n((\Delta_{\mathrm{EMStep}}S)^T\,(\Delta_{\mathrm{EMStep}}S))_n \approx S^T\left(\sum_n\left(\Delta^T_{\mathrm{EMStep}}\Delta_{\mathrm{EMStep}}\right)_n\right)S$
    15.15. $$C = S^T C_{\mathrm{EMStep}} S$$
    15.16. On the assumption that the amplification factor is constant in each of the *n* imagined iterations.

Calculating texture areas and uncertainties.

16. We now have a set of quantities and associated errors corresponding to the fraction of an image made up of each texture (5). We need to combine those that make up each terrain to get the fraction of each image covered by the terrain.
    - 16.1. We write B = KA, where B is a vector of fractional terrain areas, A is a vector of fractional texture areas, and K maps the textures onto the terrains they are part of. An element of K=1 if the texture is part of the terrain, and 0 if it is not. Textures can only be part of one terrain.
        - 16.1.1. To convert from Q to A we use…
        - 16.1.2. $$A_k = Q_k \sum_x \frac{P(x|k)a_x}{M_x} \left(= \sum_x \frac{Q_k P(x|k)}{M_x} a_x \right)$$
        - 16.1.3. Where a_x is the total area of the image covered by blobs *x* and M_x is the amount of instances of *x* in the model of the image (8.4). (The version in brackets is more readily understood. The numerator is the number of instances of *x* attributable to *k* in the model. The denominator is the number of instances of *x* in the model.)
        - 16.1.4. Since we have a covariance matrix for *Q,* we can calculate one for *A.*
        - 16.1.5. $C_A = \nabla A C_Q \nabla A^T$ where $\nabla A_{ij} = \frac{\partial A_i}{\partial Q_j} = 0 \text{ if } i \neq j$
        - 16.1.6. From 16.1.2, when *i=j*, $\nabla A_{ii} = \frac{A_i}{Q_i}$
        - 16.1.7. This converts amounts of texture into areas of texture.
    - 16.2. Terrain areas are generated by summing the areas of textures covered by the terrain (16.1).
    - 16.3. The corresponding covariance matrix is generated by $C_B = KAK^T$.
    - 16.4. The appropriateness of the treatment can be checked by inspection of χ^2 per degree of freedom.

Bolton, UK, January 2015
Lagos, Portugal, May 2015