HOW TO APPROXIMATE A NUMBER OF SEGMENTS BY A NORMAL DENSITY

JAN DE LEEUW

CONTENTS

1. BACKGROUND

Brown and Hwang [1] discuss the problem of fitting a normal density to a scaled histogram, where the histogram is scaled "so that it encloses an area of one". The problems is formulated as a least squares problem, where the integral of the squared difference between the histogram and a normal density is minimized.

We implement a similar procedure in Xlisp-Stat [2]. In Xlisp-Stat plots with lines contain a number of *linestarts*. A linestart, in two dimensions, consists of a pair of coordinates, and the index of the "next" linestart. A linestart with a nonempty next-linestart is actually a segment, because it will run from its own start coordinates to the start-coordinates of its successor. A linestart with an empty next-linestart is just there to serve as an endpoint of one or more segments.

Thus from a given scatterplot-proto, or from a graph-proto, we can extract the linestarts, and convert them to segments. In the method `:convert-linestarts`, given below, we take linestarts corresponding to non-vertical segments, and code them as quadruples with starting-x-coordinate, ending-x-coordinate, intercept, and slope.

Our problem is now to fit a normal density to the whole set of segments. These segments could come from a histogram. This is the special case of x-contiguous segments of slope zero. But they can also come from an application of `plot-function`, because functions are coded in the plot as a large number of contiguous line segments (usually 50). Or they could come from a module that draws frequency-polygons, or kernel-densities, or whatever. We can deal with overlapping segments, with non-contiguous segments, and even with arbitrary sets of segments.

Because of the generality we deal with scaling a bit differently than [1]. We have incorporated the case in which we fit a normal density to a presumably scaled histogram or to a scaled kernel-density estimate or to a function plot of, for example, a gamma density. But we have also adapted our approach in such a way that it can

deal with a free scaling parameter in the least squares loss function, over which we optimize.

Finally, there is the eternal problem how to deal with the ends of the scale. Currently, we ignore the parts of the tail in which there are no segments. In the context of histograms, it may be natural to think of the tails as semi-infinite horizontal segments with zero height. Thus, in [1], these tails add to the loss. Because we think in terms of segments actually present in the scatterplot, we do not fit the tails. But there is some arbitrariness here, because empty "interior" bins in the histogram do correspond with actual segments, and thus they contribute to the loss.

2. Formalization

We want to minimize

$$\mathcal{F}_\star(\gamma,\mu,\sigma)=\sum_{i=1}^n\mathcal{F}_i(\gamma,\mu,\sigma), \tag{2.1}$$

where

$$\mathcal{F}_i(\gamma,\mu,\sigma)=\int_{x_i}^{y_i}[(a_i+b_iz)-\gamma\phi_{\mu,\sigma}(z)]^2dz. \tag{2.2}$$

The $\{x_i,y_i,a_i,b_i\}$ are known constants, with $x_i<y_i$, while $\phi_{\mu,\sigma}(\bullet)$ is the density of a normal rv with mean μ and standard-deviation σ.

We need to minimize this expression over γ,μ and σ, but we shall also consider the case in which we minimize only over μ and σ, with γ fixed at $+1$.

In order to distinguish these cases, we introduce

$$\mathcal{G}_\star(\mu,\sigma)=\min_\gamma\mathcal{F}_\star(\gamma,\mu,\sigma), \tag{2.3}$$

$$\mathcal{H}_\star(\mu,\sigma)=\mathcal{F}_\star(1,\mu,\sigma) \tag{2.4}$$

These functions obviously have to be minimized over μ and σ.

3. Simplification

Define

$$\mathcal{A}_i(\mu,\sigma)\triangleq\int_{x_i}^{y_i}(a_i+b_iz)^2dz, \tag{3.1}$$

$$\mathcal{B}_i(\mu,\sigma)\triangleq\int_{x_i}^{y_i}(a_i+b_iz)\phi_{\mu,\sigma}(z)dz, \tag{3.2}$$

$$\mathcal{C}_i(\mu,\sigma)\triangleq\int_{x_i}^{y_i}\phi^2_{\mu,\sigma}(z)dz. \tag{3.3}$$

Let us evaluate these integrals one by one. Obviously

$$\mathcal{A}_i(\mu,\sigma)=a_i^2[y_i-x_i]+a_ib_i[y_i^2-x_i^2]+\frac{1}{3}b_i^2[y_i^3-x_i^3]. \tag{3.4}$$

Then

$$\mathcal{B}_i(\mu,\sigma)=a_i[\Phi(\frac{y_i-\mu}{\sigma})-\Phi(\frac{x_i-\mu}{\sigma})]+b_i\int_{x_i}^{y_i}z\phi_{\mu,\sigma}(z)dz, \tag{3.5}$$

with $\Phi(\bullet)$ the standard normal cdf. Also

$$\int_{x_i}^{y_i} z\phi_{\mu,\sigma}(z)dz = \mu[\Phi(\frac{y_i-\mu}{\sigma}) - \Phi(\frac{x_i-\mu}{\sigma})] - \sigma[\phi(\frac{y_i-\mu}{\sigma}) - \phi(\frac{x_i-\mu}{\sigma})], \tag{3.6}$$

from which

$$\mathcal{B}_i(\mu,\sigma) = (a_i+b_i\mu)[\Phi(\frac{y_i-\mu}{\sigma}) - \Phi(\frac{x_i-\mu}{\sigma})] - b_i\sigma[\phi(\frac{y_i-\mu}{\sigma}) - \phi(\frac{x_i-\mu}{\sigma})]. \tag{3.7}$$

To evaluate the third integral we observe that

$$\phi^2_{\mu,\sigma}(z) = \frac{1}{2\sigma\sqrt{\pi}}\phi_{\mu,\tilde{\sigma}}(z), \tag{3.8}$$

with $\tilde{\sigma} = \sigma/\sqrt{2}$. Thus

$$\mathcal{C}_i(\mu,\sigma) = \frac{1}{2\sigma\sqrt{\pi}}[\Phi(\frac{y_i-\mu}{\tilde{\sigma}}) - \Phi(\frac{x_i-\mu}{\tilde{\sigma}})]. \tag{3.9}$$

Now

$$\mathcal{F}_\star(\gamma,\mu,\sigma) = \mathcal{A}_\star(\mu,\sigma) - 2\gamma\mathcal{B}_\star(\mu,\sigma) + \gamma^2\mathcal{C}_\star(\mu,\sigma), \tag{3.10}$$

and thus

$$\mathcal{G}_\star(\mu,\sigma) = \mathcal{A}_\star(\mu,\sigma) - \frac{\mathcal{B}^2_\star(\mu,\sigma)}{\mathcal{C}_\star(\mu,\sigma)}, \tag{3.11}$$

while

$$\mathcal{H}_\star(\mu,\sigma) = \mathcal{A}_\star(\mu,\sigma) - 2\mathcal{B}_\star(\mu,\sigma) + \mathcal{C}_\star(\mu,\sigma). \tag{3.12}$$

The actual minimization is carried out using the built-in function `newtonmax`, which only uses function-values and approximates gradient and hessian by finite difference approximations. So far, these seems to work well.

References

[1] L. D. Brown and J.T. Hwang, *How to Approximate a Histogram by a Normal Density*, American Statistician, **47** (1993), 251–255.

[2] L. Tierney, *LISP-STAT. An Object-Oriented Environment for Statistical Computing and Dynamic Graphics*, Wiley, New York, 1990.

4. Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; This is inspired by Brown and Hwang, American Statistician, 1993,
;; 251-255, see also, 1994, 353-354. It is far more general, however.
;; It takes a scatterplot-proto, extracts all the linestarts, then
;; reconstructs from this all the line segments in the plot, eliminating
;; the vertical segments in the process. These segments can result from
;; a (normalized) histogram, or from a function plot, or just from
;; an arbitrary bunch of linestarts. Some lines can be horizontal,
;; some can run from right to left. Segments are coded by their
;; endpoints on the horizontal axes, intercepts, and slopes.
;;
;; The program then minimizes
;;
;; sum_{i=1}^n int_{x_i}^{y_i}(a_i+b_i z-cf(z,m,s))^2dz
;;
;; where x_i,y_i are the endpoints of segment i, a_i is the
;; intercept of the correspoding line and b_i is its slope,
;; and f(z,m,s) is the density of a normal with mean m and
;; standard-deviation s.
;;
;; We minimize either over m,s,c (default), i.e. we also
;; scale the fitted density, or we minimize over m,s fixing
;; c at 1. This can be used to fit a normal to a histogram,
;; but also to fit a normal to the plot of a gamma density,
;; and so on.
;;
;; Version 1.0 ** March 23 1995 ** Jan de Leeuw       ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defmeth scatterplot-proto :fit-normal
  (&key (mean 0) (sigma 1) (scale t) (draw t) (verbose nil))
"Args: x
Given a list of linestarts, which define a number of
segments, this fits a normal curve to the segments by minimizing mean
square error."
(let ((lst (send self :convert-linestarts))
      (arg (list mean sigma)))
  (setf arg (newtonmax #'(lambda (arg)
                           (- (normal-fit-loss-function
                               lst (first arg) (second arg) scale))) arg
                           :verbose verbose))
  (if verbose
      (format t "~,10f ~%"
              (normal-fit-loss-function lst (first arg) (second arg) scale)))
  (if draw
      (let* ((ccc (normal-fit-collect-coefs lst (first arg) (second arg)))
```

```
            (c11 (second ccc))
            (c22 (third ccc))
            (srt (mapcar #'first lst))
            (end (mapcar #'second lst)))
        (send self :add-function
              #'(lambda (x)
                  (let ((fac (if scale (/ c11 c22) 1))
                        (mn (first arg))
                        (sg (second arg)))
                    (* fac (/ (normal-dens (/ (- x mn) sg)) sg))
                    ))
              (min srt) (max end))
        (send self :adjust-to-data)) arg)
  ))

(defmeth scatterplot-proto :convert-linestarts ()
"Args: lines
The list of linestarts is converted to a list of endpoints,
slopes, and intercepts."
(let* ((nn (send self :num-lines))
       (sg nil))
  (dotimes (i nn)
    (let ((j (send self :linestart-next i)))
      (if j (let ((x0 (send self :linestart-coordinate 0 i))
                  (y0 (send self :linestart-coordinate 1 i))
                  (x1 (send self :linestart-coordinate 0 j))
                  (y1 (send self :linestart-coordinate 1 j)))
              (if (/= x0 x1)
                  (let ((a (/ (- (* y0 x1) (* y1 x0))
                              (- x1 x0)))
                        (b (/ (- y1 y0) (- x1 x0))))
                    (setf sg (append sg (list x0 x1 a b))))))))))
  (split-list sg 4)
  ))

(defun normal-fit-collect-coefs (lst mn sg)
  (let* ((srt (mapcar #'first lst))
         (end (mapcar #'second lst))
         (int (mapcar #'third lst))
         (slp (mapcar #'fourth lst))
         (c00 (sum (+ (* (^ slp 2) (/ (- (^ end 3) (^ srt 3)) 3))
                      (* slp int (- (^ end 2) (^ srt 2)))
                      (* (^ int 2) (- end srt)))))
         (df1 (/ (- srt mn) sg))
         (df2 (/ (- end mn) sg))
         (c11 (sum (- (* (+ int (* mn slp))
                         (- (normal-cdf df2)
                            (normal-cdf df1)))
```

```
                    (* sg slp
                       (- (normal-dens df2)
                          (normal-dens df1))))))
          (c22 (sum (/ (- (normal-cdf (* df2 (sqrt 2)))
                          (normal-cdf (* df1 (sqrt 2))))
                       (* 2 sg (sqrt pi))))))
     (list c00 c11 c22)
     ))

(defun normal-fit-loss-function (lst mn sg sc)
  (let* ((ccc (normal-fit-collect-coefs lst mn sg))
         (c00 (first ccc))
         (c11 (second ccc))
         (c22 (third ccc)))
    (if sc (- c00 (/ (^ c11 2) c22))
      (- (+ c00 c22) (+ c11 c11)))
    ))

(provide "fit-normal")
```

INTERDIVISIONAL PROGRAM IN STATISTICS UCLA, 8118 MATHEMATICAL SCIENCES BUILDING, 405 HILGARD AVENUE, LOS ANGELES, CA 90024
E-mail address: deleeuw@stat.ucla.edu