Square Root Rank Estimation

Shira Kritchman

Friday, July 24, 2015

The function 'sqrtFitMSE' tries to feet sqrt law to the eigenvalues distribution close to the maximal noise eigenvalue. It gets the vector of eigenvalues 'ell', and also a value 'q'. It performs the fitting assuming that the signal rank is 'q'. The function returns the MSE for the best fit it could find.

The estimation is based on the following: Let $g(x)$ be the limiting ($p, n \to \infty$, $p/n \to$ const) distribution for the noise population eigenvalues, then in the limit the noise sample eigenvalues have some limiting distribution $f(x)$. Let ℓ^* be the upperbound on the support of f, then in $[\ell^* - \epsilon, \ell^*]$ (for some ϵ) $f(x)$ behaves like $\beta_1\sqrt{\ell^* - x}$ (where β_1 is some constant depending on g).

Therefore the CDF is something like $F(x) = F(\ell^*) - \beta_2(\ell^* - x)^{3/2}$. Denote $y(x) = [F(\ell^*) - F(x)]^{2/3}$, then we get the following linear equation

$$y(x) = \beta_3(\ell^* - x)$$

And this is the equation we are trying to fit.

Denote by ℓ the vector of sample eigenvalues in descending order. We would like to find a linear fit between some values $\bar{x}$ and $\bar{y}$. Assuming the rank is q, how do we choose $\bar{x}$? Let $\ell_{\min}$ be the smallest eigenvalue which is > 0, and define the range $R = (\ell_{q+1} - m)/10$, then for a given q we take $\bar{x}$ to be all the eigenvalues in ℓ that are in $[\ell_{q+1} - r, \ell_{q+1}]$. For ℓ_i, we estimate $F(\ell_i)$ by $\frac{p-i+1}{p}$ (fraction of sample eigenvalues $\leq \ell_i$). We also estimate $F(\ell^*)$ by 1, and now we can compute $\bar{y}$ by $y = [1 - F(x)]^{2/3}$ and try to find a linear fit between $\bar{y}$ and $\bar{x}$. But we fit with constraints on the parameter ℓ^*: we require that ℓ^* is in the range $[l, L]$, where $l = (\ell_{q+1} + \ell_{q+2})/2$, and $L = (\ell_q + \ell_{q+1})/2$ (when $q = 0$ we take $L = \infty$).

```
sqrtFitMSE <- function(q, ell, d=10) {
        p <- length(ell)
        m <- min(ell[ell>0]) # smallest non-zero eigenvalue
        range <- (ell[q+1] - m) / d #we want to consider a range which is 1/d from the total range of e
        j <- which.min(ell > (ell[q+1]-range)) #we want to take eigenvalues that are between ell[q+1] a
        # we should take ell from (q+1) to j
        if (j < q+10) {j <- q+10} #we want to use at least 10 eigenvalues
        x <- ell[(q+1):j]
        u <- ((p-q):(p-j+1))/(p)
        y <- (1-u)^(2/3)
        l <- (ell[q+1] + ell[q+2]) / 2
        if (q==0) {
                L <- Inf
        } else {
                L <- (ell[q] + ell[q+1]) / 2
        }
        constrainedFit <- lm(y ~ x)
        lm <- -constrainedFit$coefficients[["(Intercept)"]] / constrainedFit$coefficients[["x"]]
        if (lm < l || lm > L) {
                b <- sum(y)/(sum(l-x))
                return(sum((y-b*x)^2)/length(x))
        }
        return(mean(resid(constrainedFit)^2))
}
```

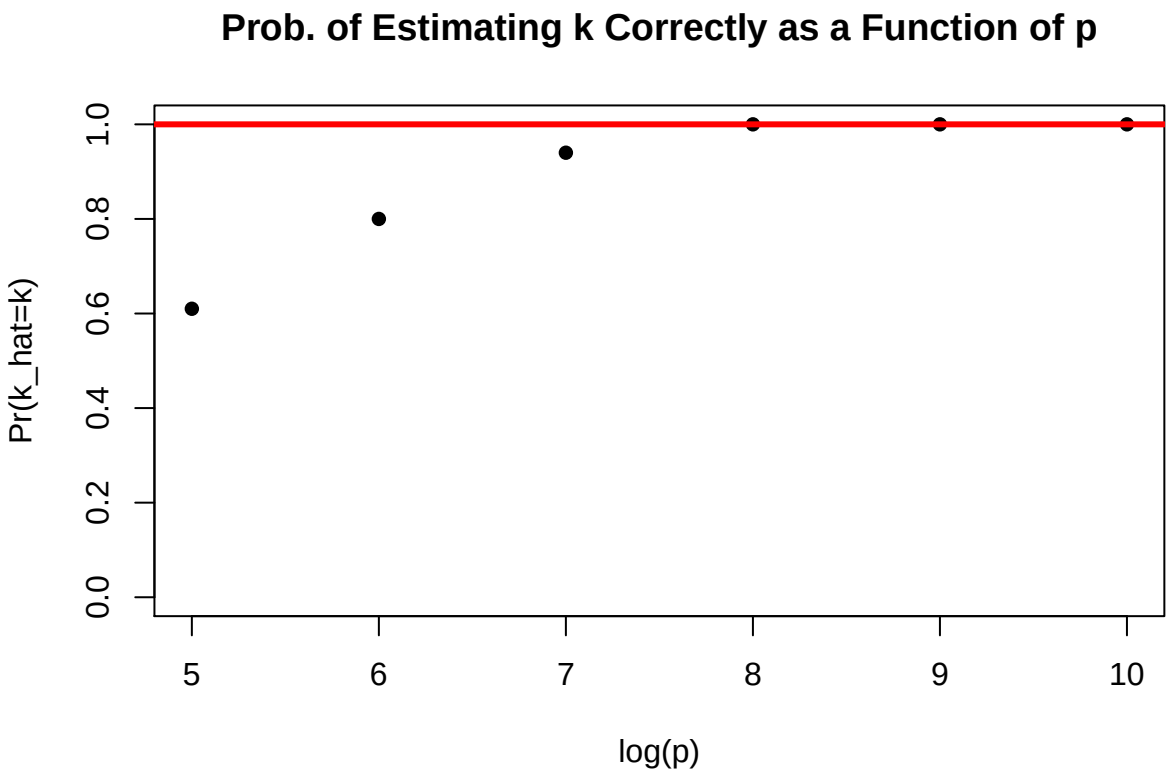
The function 'kEst' gets as input the vector of eigenvalues 'ell'. For each possible value 'q' of the rank (currently limited to q in 0:10) it uses 'sqrtFitMSE' to compute the MSE of fitting a sqrt law to the distribution of the

eigenvalues, under the assumption of 'q' signals. It then estimate the rank by taking 'q' with the minimal MSE.

```
kEst <- function(ell, k_max=10) {
        mse <- rep(0,k_max+1)
        for (q in 0:max_k) {
                mse[q+1] <- sqrtFitMSE(q, ell)
                }
        k <- which.min(mse)-1
}
```

Testing the algorithm: we test the estimator with various values for p, where $n = p$, noise eigenvalues are $U[0.5, 1.5]$, and we have rank $k = 2$ with signal strengths $\lambda = (10, 4)$. For each value of p we make 100 iterations and show the fraction of correct estimations.

```
set.seed(26580)
P <- 2^(5:10)
lam <- c(10,4)
k <- length(lam)
max_k <- 10
iter <- 100
frac <- rep(0,length(P))
for (i_P in 1:length(P)) {
        p <- P[i_P]
        n <- p
        count <- 0
        for (i_iter in 1:iter) {
                Sigma <- diag(c(lam, rep(0,p-k)) + runif(p,0.5,1.5))
                S <- rWishart(1,n,Sigma)
                S <- S[,,1]/n
                ell <- eigen(S,TRUE,TRUE)$values
                k_hat <- kEst(ell)
                if (k_hat == k) {count <- count+1}
                }
        frac[i_P] <- count/iter
        }
```


Prob. of Estimating k Correctly as a Function of p
Pr(k_hat=k)
0.0
0.2
0.4
0.6
0.8
1.0
5
6
7
8
9
10
log(p)