Rank Detection With Unknown Noise Distribution

Shira Kritchman

Sunday, July 12, 2015

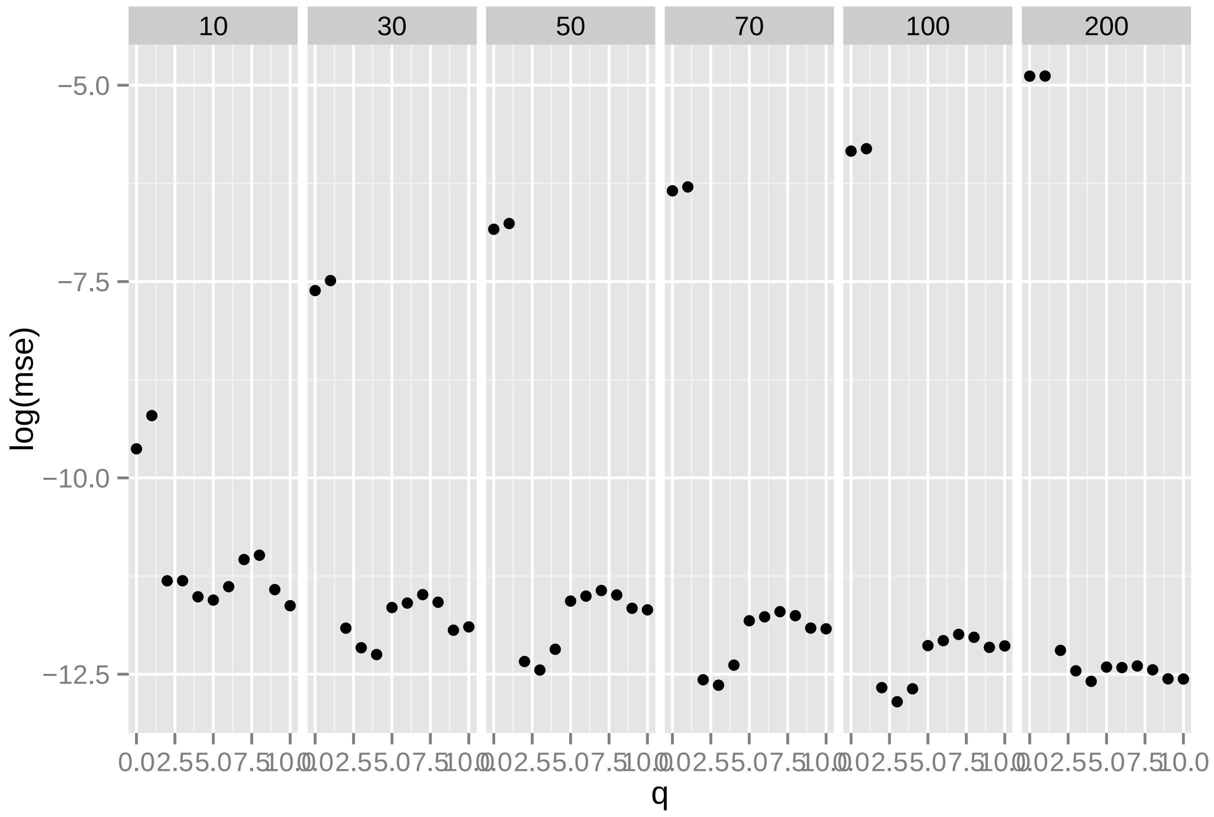
Detection of the number of signals embedded in noise is a fundamental problem in signal and array processing. In this work we consider the case where the noise covariance matrix is arbitrary and unknown and we are only given signal bearing samples. The algorithm is based on results by Dozier and Silverstein, showing that under certain conditions, the asymptotic distribution of the sample covariance matrix eigenvalues, denoted $f(x)$, behaves like $\sqrt{M-x}$ in the vicinity of M, where M is the right edge of the support of f.

Notation: * p observations dimention (number of censors etc.) * n number of samples * X is $n \times p$ matrix of samples * $S = \frac{1}{n} X^H X$ sample covariance matrix * ℓ eigenvalues of S, listed in descending order * k true rank * N size of neighborhood to fit to the sqare root law * $\lambda_1, \ldots, \lambda_k$ signa eigenvalues

The basic algorithm is given p and n and the vector ℓ. For each possibble value q of the unknown true rank k, it tries to fit a square root law to the distribution of the eigenvalues in a vicinity of ℓ_{q+1}. Since we don't know the neither the true asymptotic distribution f or the finite distribution f_n, we use the empirical CDF. Note that if $f(x) \approx c_1\sqrt{M-x}$ then the CDF is $F(x) \approx 1 - c_2(M-x)^{1.5}$. So for a given q we try to fit $1 - c_2(M-x)^{1.5}$ to $\ell_{q+1}, \ldots, \ell_{q+N}$, where the value of N will be discussed later. Since the p eigenvalues of S are listed in descending order, the empirical CDF of ℓ_i is $(p-i+1)/p$. So we want to find x_2 and M such that $i/p = 1 - c_2(M-\ell_i)^1.5$ for $i = q+1, \ldots, q+R$. Denote $y_i = (1-i/p)^{2/3}$, then we have $c_3 - c_2\ell_i = y_i$. Therefore we look for a linear fit of ell_i with y_i. For each possible value q of k we compute the MSE of the best linear fit. Here is the function computing this value:

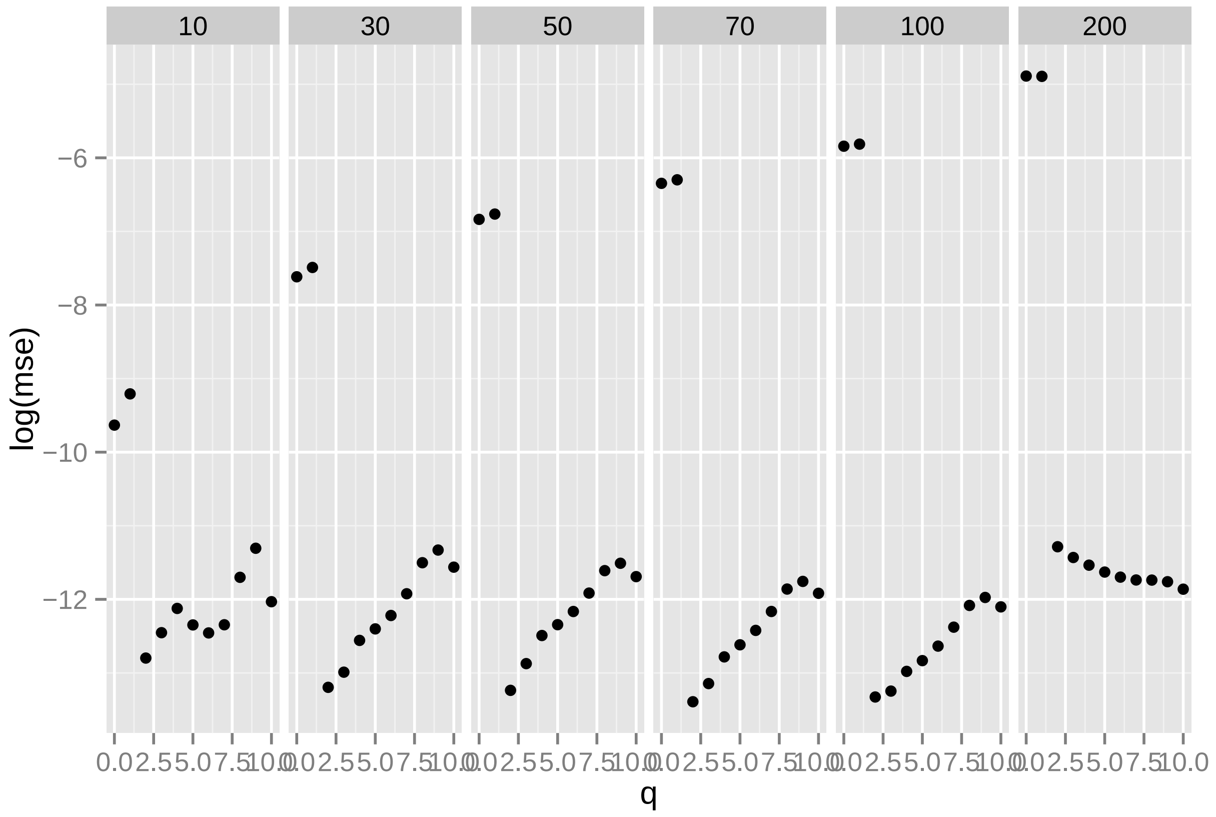
```
sqrtFitMSE <- function(q, ell, N, applyWeights=FALSE) {
        p <- length(ell)
        x <- ell[(q+1):(q+N)]
        u <- ((p-q):(p-q-N+1))/(p-q)
        y <- (1-u)^(2/3)
#         if (applyWeights==FALSE) {
#                 w <- NULL
#                 } else {
#                         w <- 1/(1+y)
#                         w <- (length(x):1)^2
#                         }
        fit <- lm(y~x)
        return(mean(fit$residuals^2))
        }
```

For example take Gaussian white noise with $\sigma^2 = 1$, take $p = 1000$, $n = 2000$, and take $k = 2$ with $\lambda = (100, 50)$. Then we expect to see the sqrt fit MSE at a minimum for $q = 2$. Here is a plot of $sqrtFitMSE$ for simulated data and with different values for N:



On the one hand, in this simple scenario, we get a clear separation of the MSE between $q \geq k$ and $q < k$. On the other hand, the minimal MSE is not promised to be at $q = k$, so we are not sure how to set the predictor $\hat{k}$.

Now we do the same experiment, only with different noise. Noise covariance matrix eigenvalues are now $\sim U[0.5, 1.5]$. This is how the MSE behaves for different values of N:

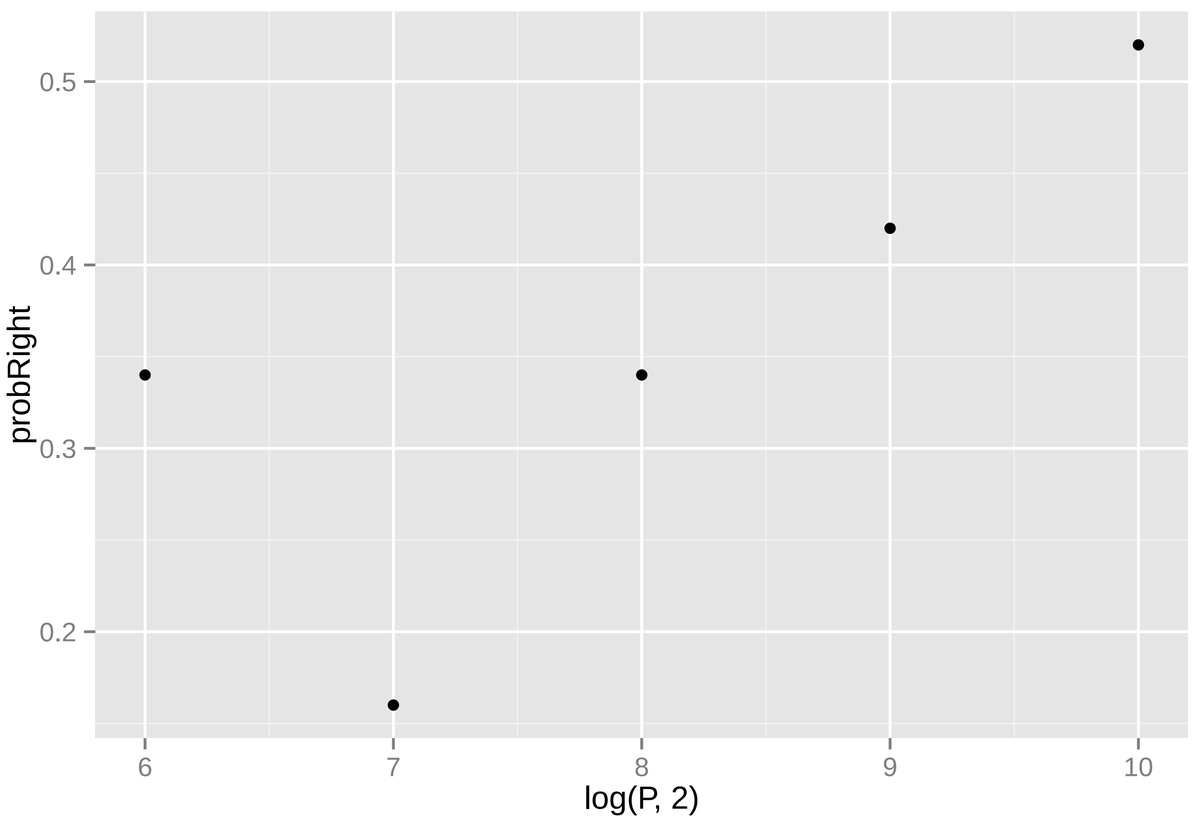


We now try to estimate k by taking q with the lowest MSE.

```
sqrtFit <- function(max_k, ell, N) {
        mse <- rep(0, max_k+1)
        for (q in 0:max_k) {
                mse[q+1] <- sqrtFitMSE(q, ell, N, applyWeights=FALSE)
                }
        return(which.min(mse)-1)
        }
```

```
iter <- 50
```

We apply this estimator with different values of p, with $n = 2 * p$. For N we use $\sqrt{p}$. For each value of p we make 50 iterations. We show the probability of getting k right for each value of p.


0.5
0.4
0.3
0.2
probRight
6
7
8
9
10
log(P, 2)