Diogo Melo

Exercícios

1. Exercício 1
2. Exercício 2
3. Exercício 3
4. Exercício 4
5. Exercício 5
6. Exercício 6
7. Exercício 7
8. Exercício 8

Projeto Final

Proposta A

Implmentar algoritmos para calcular a correção de Stein em um vetor de dados amostrais. Se dois ou mais vetores forem dados, calcular a matriz de covariância entre os vetores com a correção de Stein.

Algumas referências: Effron & Moris 1977 ou ainda Schäfer & Strimmer 2005

Comentários PI

Ótimo, e obrigado pelos artigos. É possível incluir alguma informação sobre o quão diferentes são os dados ou a estatísticas originais e corrigidos? Por exemplo, vc pode retornar as amtrizes de covariâncias com e sem correção e algumas métricas de comparação entre elas, para que o usuário avalie o que está ganhando(ou perdendo) com a correção.

Comentário do comentário PI

Uma medida legal é olhar a distribuição dos auto-valores antes e depois da correção. Não sei se tem alguma métrica pra comparar as matrizes diretamente. Outra coisa possível é alterar as medidas originais de modo que elas resultem na matriz corrigida. Assim vc pode comprar na escala das medidas o quanto de alterações vc está fazendo.

Plano B (1.1)

Criar uma função para visualização de matrizes de covariância em pseudo-cor e alguns gráficos diagnostico, como distribuição das correlações, distribuição dos auto-valores, primeiros componentes principais e percentual de variação explicados por eles.

Comentários PI

Parece interessante, mas não está claro o suficiente para avaliar.

Stein

Função

```
Stein  <-  function(y)
{
   y = as.matrix(y)
   n = nrow(y)
   p = ncol(y)
   if(p>1){
      x = apply(y, 2, mean)
      w = array(dim=(c(n, p, p)))
      w.mean=array(dim=c(p,p))
      var.s=array(dim=c(p,p))
      s=array(dim=c(p,p))
      for (k in 1:n){
         for (i in 1:p){
            for (j in 1:p){
               w[k,i,j] = (y[k,i] - x[i])*(y[k,j] - x[j])
            }
         }
      }
      w.mean=array(dim=c(p,p))
      for (i in 1:p){
         for (j in 1:p){
            w.mean[i,j] = sum(w[,i,j])/n
         }
      }
      s = w.mean*n/(n-1)
      for (i in 1:p){
         for (j in 1:p){
            var.s[i,j] = sum((w[,i,j] - w.mean[i,j])*(w[,i,j] -
w.mean[i,j]))*n/((n-1)*(n-1)*(n-1))
         }
      }
```

```
      sum.var = sum(var.s) - sum(diag(var.s))
      sum.s2 = sum(s*s) - sum(diag(s)*diag(s))
      lamb = sum.var/sum.s2
      if (lamb > 1){lamb = 1}
      if (lamb < 0){lamb = 0}
      s.star = s*(1-lamb)
      s.star[row(s.star)==col(s.star)] = s[row(s)==col(s)]
      var.y = s.star
   }
   else{
      n = 1
      p = length(y)
      x = y;
      var.y = diag(var(y)[1,1], p)
      s = var(y)[1,1]
      s.star = s
   }
   eigen.y = eigen(var.y)
   eVal = eigen.y$values
   eVec = eigen.y$vectors
   target = rep(mean(y), p)
   p.true = sum(eVal)/max(eVal)
   shrink.y =  1 - (p.true - 2)/sum((x-target)*solve(var.y, x-target))
   if(shrink.y > 0){
      JS.y = target + shrink.y*(x-target)
   }
   else{
     JS.y = x
   }
   if(p>7 & n>1){
   eigen.y = eigen(s)
   eVal = eigen.y$values
   eVec = eigen.y$vectors
   grad = array(dim=c(p-2))
   tr.y = sum(eVal)
   for (i in 1:(p-2))
      grad[i] = abs(eVal[i]/tr.y - 2*(eVal[i+1]/tr.y) + eVal[i+2]/tr.y)
   var.grad = array(dim=c(p-6))
   for(i in 1:(p-6)){
      var.grad[i] = var(grad[i:(i+4)])
   }
   length(var.grad[var.grad<1e-4])
   x11()
   plot(4:(p-3),var.grad)
   corte = floor(locator(1)$x)
   eVal[eVal<eVal[corte]] = eVal[corte]
   Ext.covar = eVec%*%diag(eVal)%*%t(eVec)
   }
   else{Ext.covar= NA}
   names(JS.y) = colnames(y)
   names(x) = colnames(y)
```

```
   colnames(s) = colnames(y)
   rownames(s) = colnames(y)
   colnames(s.star) = colnames(y)
   rownames(s.star) = colnames(y)
   OutPut = list(x, JS.y, s, s.star, Ext.covar)
   names(OutPut) = c('ML', 'JS', 'ML.covar', 'JS.covar', 'Ext.covar')
   return(OutPut)
}
```

Help

```
 Stein                package:ogropacks                R Documentation


 Description:

    Calcula matrizes de covariância de um conjunto de dados usando
    máxima verossimilhança, o estimador de Stein e usando o médoto de
    extensão de auto-valores. Além disso calcula os estimadores de máxima
verossimilhança
    e de Stein para a média da distribuição normal multi-variada que
supostamente
    gerou as observações.

Usage:

    Stein(x)


Arguments:

     x: Uma matriz ou data frame cujas linhas são formadas por observações
das váriaveis de
    interesse. As correções de Stein só são admissiveis para vetores de
observações com mais de
    3 dimensões. Caso apenas uma observação de cada variável esteja
disponivel o vetor passado
    para a função deve ter uma linha e tantas colunas quantas forem as
variáveis. Neste caso
    as correções de Stein e de extensão para a matriz de covariância não se
aplicam.

Details:

     Para o método de extensão um ponto de corte dos auto-valores deve ser
selecionado. Para tal um gráfico da
    variância do gradiente dos auto-valores será apresentado ao usuário. O
ponto de corte deve ser escolhido como
```

```
    o ponto de início de platô próximo de zero desse gráfico. Basta clicar
no ponto desejado na janela gráfica.
    O método de extensão só será utilizado para dados com dimensionalidade
maior que 7 e pelo menos duas observações.


Value:

     Lista

   ML : Estimador de máxima verossimilhança ara o parametro média da
distribuição multi-variada normal que
   gerou os dados. Equivalente a média amostral. No caso de apenas uma
observação é igual ao valor da observação.

  JS : Estimador de James-Stein para o parametro média da distribuição
multi-variada normal que gerou os dados.

  ML.covar : Estimador de máxima verossimilhança para a matriz de
covariância dos dados.

  JS.covar : Estimador de James-Stein para a matriz de covariância. Exige
que exista mais de uma observação para
  cada parâmetro.

  Ext.covar : Matriz de covariância corrigida pelo método de extensão.
Recomenda-se usar esta estimativa apenas
  para dados de dimensionalidade alta (acima de 15 variáveis observadas), o
valor mínimo para essa função é 8 variáveis.


Author(s):

     Diogo Melo

References:

    'Inadmissibility of the Usual Estimator for the Mean of a Multivariate
Normal Distribution', Charles Stein, 1956
    'Honey, I Shrunk the Sample Covariance Matrix', Olivier Ledoit & Michael
Wolf, 2003.
    'A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and
Implications for Functional Genomics', Juliane Schafer & Korbinian Strimmer.
2005
    http://en.wikipedia.org/wiki/James–Stein_estimator
    'NOISE, MODULARITY AND THE PROBLEM OF THE USEFUL RANK IN MATRIX
INVERSION: AN EXAMPLE OF SELECTION RECONSTRUCTION IN NEW WORLD MONKEYS'
Gabriel Marroig & Diogo Melo, em preparação.


Examples:
```

```
 ##Exemplo com baixa dimensionalidade

 Stein(iris[,1:4])

 ## Exemplo com baixa amostragem

 Stein(rnorm(15))

 ## Exemplo com alta dimensionalidade (101 paremetros), amostragem baixa (10
observações)...

 teta = -50:50 ## Vetor de médias da distribuição N(teta, I)
 y = matrix(rnorm(10*length(teta),teta, 1), ncol = length(teta), byrow=T)
 Out=Stein(y)
 norm = function (x) {sqrt(x%*%x)}
 SR.ML = norm(Out$ML-(teta))^2
 cat('Erro quadrado da estimativa ML', SR.ML, '\n')
 SR.JS = norm(Out$JS-(teta))^2
 cat('Erro quadrado da estimativa JS', SR.JS, '\n')
 eVal.ML<-eigen(var(Out$ML.covar), only.values=T)$values
 eVal.JS<-eigen(var(Out$JS.covar), only.values=T)$values
 eVal.Ext<-eigen(var(Out$Ext.covar), only.values=T)$values
 par(mfrow=c(2,3), pty = "s")
 plot(eVal.ML, main='ML', col='red')
 points(eVal.JS, main='JS', col='blue', pch=8)
 points(eVal.Ext, main='Ext', col='green', pch=9)
 #plotando as matrizes de COVARIAÇÃO!
 PlotCov = function(x, main=''){
     n = nrow(x)
     Corr = array(dim=c(n,n))
     for( i in 1:n){
       for(j in 1:n){
          Corr[(n-i+1),j] = x[i,j]/sqrt((x[i,i]*x[j,j]))
       }
     }
     image(Corr, col=heat.colors(floor(length(Corr)/2)), main=main)
 }
 PlotCov(diag(1, length(teta)), main='Matriz Geradora')

 PlotCov(Out$ML.covar, main='ML.Covar')

 PlotCov(Out$JS.covar, main='JS.Covar')

 PlotCov(Out$Ext.covar, main='Ext.Covar')
```

Stein.r

Stein.help