Package: rpca (via r-universe)

October 11, 2024

Type Package

Title RobustPCA: Decompose a Matrix into Low-Rank and Sparse Components, truncated version, with additional L2 noise separation option

Version 0.3.2

Date 2018-01-11

Author Maciek Sykulski [aut, cre], Krzysztof Gogolewski [aut]

Maintainer Maciek Sykulski <`maciekskgmail.com`>

Description Suppose we have a data matrix, which is the superposition of a low-rank component and a sparse component. Candes, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 11. prove that we can recover each component individually under some suitable assumptions. It is possible to recover both the low-rank and the sparse components exactly by solving a very convenient convex program called Principal Component Pursuit; among all feasible decompositions, simply minimize a weighted combination of the nuclear norm and of the L1 norm. This package implements this decomposition algorithm resulting with Robust PCA approach.

License GPL-2 | GPL-3

Imports compiler, irlba, Matrix

Repository https://macieksk.r-universe.dev

RemoteUrl https://github.com/macieksk/rpca

RemoteRef HEAD

RemoteSha 197df1b727818ccd920db04d03e9d2a5446596ec

Contents

`rpca-package` *RobustPCA: Decompose a Matrix into Low-Rank and Sparse Components, truncated version, with additional L2 noise separation option*

Description

Suppose we have a data matrix, which is the superposition of a low-rank component and a sparse component. Candes, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 11. prove that we can recover each component individually under some suitable assumptions. It is possible to recover both the low-rank and the sparse components exactly by solving a very convenient convex program called Principal Component Pursuit; among all feasible decompositions, simply minimize a weighted combination of the nuclear norm and of the L1 norm. This package implements this decomposition algorithm resulting with Robust PCA approach.

Details

Index: This package was not yet installed at build time.

This package contains `rpca` function, which decomposes a rectangular matrix M into a low-rank component, and a sparse component, by solving a convex program called Principal Component Pursuit:

$$\begin{aligned} \text{minimize} \quad & \|L\|_* + \lambda \|S\|_1 \\ \text{subject to} \quad & L + S = M \end{aligned}$$

where $\|L\|_*$ is the nuclear norm of L (sum of singular values).

Note

Use `citation("rpca")` to cite this R package.

Author(s)

Maciek Sykulski [aut, cre], Krzysztof Gogolewski [aut]

Maintainer: Maciek Sykulski <macieksk@gmail.com>

References

Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 11.

Yuan, X., & Yang, J. (2009). Sparse and low-rank matrix decomposition via alternating direction methods. preprint, 12.

See Also

`rpca`

F2norm — *Frobenius norm of a matrix*

Description

Frobenius norm of a matrix.

Usage

```
F2norm(M)
```

Arguments

`M`	A matrix.

Value

Frobenius norm of M.

Examples

```
## The function is currently defined as
function (M)
sqrt(sum(M^2))

F2norm(matrix(runif(100),nrow=5))
```

---

rpca — *Decompose a matrix into a low-rank component and a sparse component by solving Principal Components Pursuit*

Description

This function decomposes a rectangular matrix *M* into a low-rank component, and a sparse component, by solving a convex program called Principal Component Pursuit.

Usage

```
rpca(M,
     lambda = 1/sqrt(max(dim(M))), mu = prod(dim(M))/(4 * sum(abs(M))),
     term.delta = 10^(-7), max.iter = 5000, trace = FALSE,
     thresh.nuclear.fun = thresh.nuclear, thresh.l1.fun = thresh.l1,
     F2norm.fun = F2norm)
```

Arguments

`M`	a rectangular matrix that is to be decomposed into a low-rank component and a sparse component $M = L + S$.
`lambda`	parameter of the convex problem $\|L\|_* + \lambda\|S\|_1$ which is minimized in the Principal Components Pursuit algorithm. The default value is the one suggested in Candès, E. J., section 1.4, and together with reasonable assumptions about L and S guarantees that a correct decomposition is obtained.
`mu`	parameter from the augumented Lagrange multiplier formulation of the PCP, Candès, E. J., section 5. Default value is the one suggested in references.
`term.delta`	The algorithm terminates when $\|M - L - S\|_F \le \delta\|M\|_F$ where $\| \ \|_F$ is Frobenius norm of a matrix.
`max.iter`	Maximal number of iterations of the augumented Lagrange multiplier algorithm. A warning is issued if the algorithm does not converge by then.
`trace`	Print out information with every iteration.
`thresh.nuclear.fun`, `thresh.l1.fun`, `F2norm.fun`	Arguments for internal use only.

Details

These functions decompose a rectangular matrix M into a low-rank component, and a sparse component, by solving a convex program called Principal Component Pursuit:

$$\text{minimize} \quad \|L\|_* + \lambda\|S\|_1$$

$$\text{subject to} \quad L + S = M$$

where $\|L\|_*$ is the nuclear norm of L (sum of singular values).

Value

The function returns two matrices `S` and `L`, which have the property that $L + S \simeq M$, where the quality of the approximation depends on the argument `term.delta`, and the convergence of the algorithm.

`S`	The sparse component of the matrix decomposition.
`L`	The low-rank component of the matrix decomposition.
`L.svd`	The singular value decomposition of `L`, as returned by the function `La.svd` .
`convergence$converged`	`TRUE` if the algorithm converged with respect to `term.delta`.
`convergence$iterations`	Number of performed iterations.
`convergence$final.delta`	The final iteration `delta` which is compared with `term.delta`.
`convergence$all.delta`	All `delta` from all iterations.

Author(s)

Maciek Sykulski [aut, cre], Krzysztof Gogolewski [aut]

References

Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 11.

Yuan, X., & Yang, J. (2009). Sparse and low-rank matrix decomposition via alternating direction methods. preprint, 12.

Examples

```
data(iris)
M <- as.matrix(iris[,1:4])
Mcent <- sweep(M,2,colMeans(M))

res <- rpca(Mcent)

## Check convergence and number of iterations
with(res$convergence,list(converged,iterations))
## Final delta F2 norm divided by F2norm(Mcent)
with(res$convergence,final.delta)

## Check properites of the decomposition
with(res,c(
all(abs( L+S - Mcent ) < 10^-5),
all( L == L.svd$u%*%(L.svd$d*L.svd$vt) )
))
# [1] TRUE TRUE

## The low rank component has rank 2
length(res$L.svd$d)
## However, the sparse component is not sparse
## - thus this data set is not the best example here.
mean(res$S==0)

## Plot the first (the only) two principal components
## of the low-rank component L
rpc<-res$L.svd$u%*%diag(res$L.svd$d)
plot(jitter(rpc[,1:2],amount=.001),col=iris[,5])

## Compare with classical principal components
pc <- prcomp(M,center=TRUE)
plot(pc$x[,1:2],col=iris[,5])
points(rpc[,1:2],col=iris[,5],pch="+")

## "Sparse" elements distribution
plot(density(abs(res$S),from=0))
curve(dexp(x,rate=1/mean(abs(res$S))),add=TRUE,lty=2)

## Plot measurements against measurements corrected by sparse components
```

```
par(mfcol=c(2,2))
for(i in 1:4) {
plot(M[,i],M[,i]-res$S[,i],col=iris[,5],xlab=colnames(M)[i])
}
```

---

thresh.l1 *Shrinkage operator*

Description

Shrinkage operator: S[x] = sgn(x) max(|x| - thr, 0). For description see section 5 of Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?.

Usage

```
thresh.l1(x, thr)
```

Arguments

`x`	a vector or a matrix.
`thr`	threshold >= 0 to shrink with.

Value

S[x] = sgn(x) max(|x| - thr, 0)

References

Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 11

Yuan, X., & Yang, J. (2009). Sparse and low-rank matrix decomposition via alternating direction methods. preprint, 12.

See Also

`thresh.nuclear`

Examples

```
## The function is currently defined as
function(x,thr){sign(x)*pmax(abs(x)-thr,0)}

summary(thresh.l1(runif(100),0.3))
```

`thresh.nuclear` *Thresholding operator*

Description

Thresholding operator, an application of the shrinkage operator on a singular value decomposition: D[X] = U S[Sigma] V . For description see section 5 of Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?.

Usage

```
thresh.nuclear(M, thr)
```

Arguments

`M`	a rectangular matrix.
`thr`	threshold >= 0 to shrink singular values with.

Value

Returned is a thresholded Singular Value Decomposition with `thr` subtracted from singular values, and values smaller than 0 dropped together with their singular vectors.

`u`, `d`, `vt`	as in return value of `La.svd`
`L`	the resulting low-rank matrix: $L = UDV^t$

References

Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 11

Yuan, X., & Yang, J. (2009). Sparse and low-rank matrix decomposition via alternating direction methods. preprint, 12.

See Also

`thresh.l1`

Examples

```
## The function is currently defined as
function (M, thr) {
    s <- La.svd.cmp(M)
    dd <- thresh.l1(s$d, thr)
    id <- which(dd != 0)
    s$d <- dd[id]
    s$u <- s$u[, id, drop = FALSE]
    s$vt <- s$vt[id, , drop = FALSE]
    s$L <- s$u %*% (s$d * s$vt)
    s
  }

l<-thresh.nuclear(matrix(runif(600),nrow=20),2)
l$d
```

---

trpca *TODO Decompose a matrix into a low-rank component and a sparse component by solving Principal Components Pursuit*

Description

TODO This function decomposes a rectangular matrix M into a low-rank component, and a sparse component, by solving a convex program called Principal Component Pursuit.

Usage

```
trpca(M,     #k,
             k.start=1,
             lambda = 1/sqrt(max(dim(M))), #This is ok only for dense matrices
        lambda2 = 100*lambda, #TODO needs proper L1 sparse vs L2 noise weight setting
             L2noise = TRUE, #Do decompose into M=L+S+E, or just M=L+S if FALSE
         mu = prod( dim(M)) / (4*sum(abs(M)) ), #This is ok only for dense matrices
             mu.max = mu*100,  #Stops mu from getting to large too fast
                     #(i.e. from caring too much for constraint than objective.function)
        mu.min = mu/200,  #If smallest computed SV is larger than 1/mu.min we increase k.current
                          #and compute one more SV in next iteration
             mu.growth.ratio=1.1,
             term.delta=10^(-7),
             max.iter=5000,
             trace=FALSE,
             message.iter=100,
        n.iter.without.L2noise=5, #Number of start iterations without decomposing L2noise
             #thresh.nuclear.fun=trpca.thresh.nuclear.sparse2,
             #thresh.l1.fun=thresh.l1.sparse,
             #zero.matrix.fun=zero.matrix.sparse,
             thresh.nuclear.fun=trpca.thresh.nuclear,
             thresh.l1.fun=thresh.l1,
             zero.matrix.fun=zero.matrix,
             F2norm.fun=F2norm)
```

Arguments

`M`	a rectangular matrix that is to be decomposed into a low-rank component and a sparse component $M = L + S$.

`lambda`	parameter of the convex problem $\|L\|_* + \lambda\|S\|_1$ which is minimized in the Principal Components Pursuit algorithm. The default value is the one suggested in Candès, E. J., section 1.4, and together with reasonable assumptions about L and S guarantees that a correct decomposition is obtained.
`mu`	parameter from the augumented Lagrange multiplier formulation of the PCP, Candès, E. J., section 5. Default value is the one suggested in references.
`term.delta`	The algorithm terminates when $\|M - L - S\|_F \leq \delta\|M\|_F$ where $\| \ \|_F$ is Frobenius norm of a matrix.
`max.iter`	Maximal number of iterations of the augumented Lagrange multiplier algorithm. A warning is issued if the algorithm does not converge by then.
`trace`	Print out information with every iteration.
`thresh.nuclear.fun`, `thresh.l1.fun`, `F2norm.fun`	Arguments for internal use only.

Details

TODO, documentation for original rpca below.

These functions decompose a rectangular matrix M into a low-rank component, and a sparse component, by solving a convex program called Principal Component Pursuit:

$$\text{minimize} \quad \|L\|_* + \lambda\|S\|_1$$

$$\text{subject to} \quad L + S = M$$

where $\|L\|_*$ is the nuclear norm of L (sum of singular values).

Value

The function returns two matrices `S` and `L`, which have the property that $L + S \simeq M$, where the quality of the approximation depends on the argument `term.delta`, and the convergence of the algorithm.

`S`	The sparse component of the matrix decomposition.
`L`	The low-rank component of the matrix decomposition.
`L.svd`	The singular value decomposition of `L`, as returned by the function `La.svd` .
`convergence$converged`	`TRUE` if the algorithm converged with respect to `term.delta`.
`convergence$iterations`	Number of performed iterations.
`convergence$final.delta`	The final iteration `delta` which is compared with `term.delta`.
`convergence$all.delta`	All `delta` from all iterations.

Author(s)

Maciek Sykulski [aut, cre], Krzysztof Gogolewski [aut]

References

Candès, E. J., Li, X., Ma, Y., & Wright, J. (2011). Robust principal component analysis?. Journal of the ACM (JACM), 58(3), 11.

Yuan, X., & Yang, J. (2009). Sparse and low-rank matrix decomposition via alternating direction methods. preprint, 12.

Examples

```
## TODO original rpca examples below

data(iris)
M <- as.matrix(iris[,1:4])
Mcent <- sweep(M,2,colMeans(M))

res <- rpca(Mcent)

## Check convergence and number of iterations
with(res$convergence,list(converged,iterations))
## Final delta F2 norm divided by F2norm(Mcent)
with(res$convergence,final.delta)

## Check properites of the decomposition
with(res,c(
all(abs( L+S - Mcent ) < 10^-5),
all( L == L.svd$u%*%(L.svd$d*L.svd$vt) )
))
# [1] TRUE TRUE

## The low rank component has rank 2
length(res$L.svd$d)
## However, the sparse component is not sparse
## - thus this data set is not the best example here.
mean(res$S==0)

## Plot the first (the only) two principal components
## of the low-rank component L
rpc<-res$L.svd$u%*%diag(res$L.svd$d)
plot(jitter(rpc[,1:2],amount=.001),col=iris[,5])

## Compare with classical principal components
pc <- prcomp(M,center=TRUE)
plot(pc$x[,1:2],col=iris[,5])
points(rpc[,1:2],col=iris[,5],pch="+")

## "Sparse" elements distribution
plot(density(abs(res$S),from=0))
curve(dexp(x,rate=1/mean(abs(res$S))),add=TRUE,lty=2)

## Plot measurements against measurements corrected by sparse components
par(mfcol=c(2,2))
for(i in 1:4) {
```

```
plot(M[,i],M[,i]-res$S[,i],col=iris[,5],xlab=colnames(M)[i])
}
```

Index