Package ‘tvR’

October 27, 2019

Type Package

Title Total Variation Regularization

Version 0.3.1

Description Provides tools for denoising noisy signal and images via Total Variation Regularization. Reducing the total variation of the given signal is known to remove spurious detail while preserving essential structural details. For the seminal work on the topic, see Rudin et al (1992) <doi:10.1016/0167-2789(92)90242-F>.

License GPL (>= 3)

Encoding UTF-8

LazyData true

Depends R (>= 2.14.0)

Imports Rcpp, Matrix, Rdpack, utils

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 6.1.1

RdMacros Rdpack

URL http://github.com/kyoustat/tvR

BugReports http://github.com/kyoustat/tvR/issues

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Kisung You [aut, cre] (<https://orcid.org/0000-0002-8584-459X>)

Maintainer Kisung You <kyoustat@gmail.com>

Repository CRAN

Date/Publication 2019-10-27 17:10:02 UTC

R topics documented:

tvR-package 2
denoise1 2
denoise2 4
lena128 6

Index **7**

`tvR-package` *tvR : Total Variation Regularization*

Description

tvR provides tools for denoising noisy signal and images via Total Variation Regularization. Reducing the total variation of the given signal is known to remove spurious detail while preserving essential structural details. For now, we provide tools for *denoising* only on 1-dimensional signals or 2-dimensional images, where the latter be represented as 2d or 3d array.

`denoise1` *Total Variation Denoising for Signal*

Description

Given a 1-dimensional signal `f`, it solves an optimization of the form,

$$u^* = argmin_u E(u, f) + \lambda V(u)$$

where $E(u, f)$ is fidelity term and $V(u)$ is total variation regularization term. The naming convention of a parameter `method` is `<problem type>` + `<name of algorithm>`. For more details, see the section below.

Usage

```
denoise1(signal, lambda = 1, niter = 100, method = c("TVL2.IC",
  "TVL2.MM"))
```

Arguments

`signal`	vector of noisy signal.
`lambda`	regularization parameter (positive real number).
`niter`	total number of iterations.
`method`	indicating problem and algorithm combination.

Value

a vector of same length as input `signal`.

Algorithms for TV-L2 problem

The cost function for TV-L2 problem is

$$min_u \frac{1}{2}|u - f|_2^2 + \lambda|\nabla u|$$

where for a given 1-dimensional vector, $|\nabla u| = \sum |u_{i+1} - u_i|$. Algorithms (in conjunction with model type) for this problems are

`"TVL2.IC"` Iterative Clipping algorithm.

`"TVL2.MM"` Majorization-Minorization algorithm.

The codes are translated from MATLAB scripts by Ivan Selesnick.

References

Rudin LI, Osher S, Fatemi E (1992). “Nonlinear total variation based noise removal algorithms.” *Physica D: Nonlinear Phenomena*, **60**(1-4), 259–268. ISSN 01672789, doi: 10.1016/01672789(92)90242-F, `http://linkinghub.elsevier.com/retrieve/pii/016727899290242F`.

Selesnick IW, Parekh A, Bayram I (2015). “Convex 1-D Total Variation Denoising with Non-convex Regularization.” *IEEE Signal Processing Letters*, **22**(2), 141–144. ISSN 1070-9908, 1558-2361, doi: 10.1109/LSP.2014.2349356, `http://ieeexplore.ieee.org/document/6880761/`.

Examples

```
## generate a stepped signal
x = rep(sample(1:5,10,replace=TRUE), each=50)

## add some additive white noise
xnoised = x + rnorm(length(x), sd=0.25)

## apply denoising process
xproc1 = denoise1(xnoised, method = "TVL2.IC")
xproc2 = denoise1(xnoised, method = "TVL2.MM")

## plot noisy and denoised signals
plot(xnoised, pch=19, cex=0.1, main="Noisy signal")
lines(xproc1, col="blue", lwd=2)
lines(xproc2, col="red", lwd=2)
legend("bottomleft",legend=c("Noisy","TVL2.IC","TVL2.MM"),
col=c("black","blue","red"),#' lty = c("solid", "solid", "solid"),
lwd = c(0, 2, 2), pch = c(19, NA, NA),
pt.cex = c(1, NA, NA), inset = 0.05)
```

denoise2 *Total Variation Denoising for Image*

Description

Given an image `f`, it solves an optimization of the form,

$$u^* = argmin_u E(u,f) + \lambda V(u)$$

where $E(u,f)$ is fidelity term and $V(u)$ is total variation regularization term. The naming convention of a parameter `method` is `<problem type>` + `<name of algorithm>`. For more details, see the section below.

Usage

```
denoise2(data, lambda = 1, niter = 100, method = c("TVL1.PrimalDual",
  "TVL2.PrimalDual", "TVL2.FiniteDifference"), normalize = FALSE)
```

Arguments

`data`	standard 2d or 3d array.
`lambda`	regularization parameter (positive real number).
`niter`	total number of iterations.
`method`	indicating problem and algorithm combination.
`normalize`	a logical; `TRUE` to make the range in $[0,1]$, or `FALSE` otherwise.

Value

denoised array as same size of `data`.

Data format

An input `data` can be either (1) 2-dimensional matrix representaing *grayscale* image, or (2) 3-dimensional array for *color* image.

Algorithms for TV-L1 problem

The cost function for TV-L2 problem is

$$min_u |u-f|_1 + \lambda |\nabla u|$$

where for a given 2-dimensional array, $|\nabla u| = \sum sqrt(u_x^2 + u_y^2)$ Algorithms (in conjunction with model type) for this problems are

`"TVL1.PrimalDual"` Primal-Dual algorithm.

Algorithms for TV-L2 problem

The cost function for TV-L2 problem is

$$min_u |u - f|_2^2 + \lambda |\nabla u|$$

and algorithms (in conjunction with model type) for this problems are

`"TVL2.PrimalDual"` Primal-Dual algorithm.

`"TVL2.FiniteDifference"` Finite Difference scheme with fixed point iteration.

References

Rudin LI, Osher S, Fatemi E (1992). "Nonlinear total variation based noise removal algorithms." *Physica D: Nonlinear Phenomena*, **60**(1-4), 259–268. ISSN 01672789, doi: 10.1016/01672789(92)90242-F, `http://linkinghub.elsevier.com/retrieve/pii/016727899290242F`.

Chambolle A, Pock T (2011). "A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging." *Journal of Mathematical Imaging and Vision*, **40**(1), 120–145. ISSN 0924-9907, 1573-7683, doi: 10.1007/s1085101002511, `http://link.springer.com/10.1007/s10851-010-0251-1`.

Examples

```
## Not run:
## Load grey-scale 'lena' data
data(lena128)

## Add white noise
sinfo   <- dim(lena128)   # get the size information
xnoised <- lena128 + array(rnorm(128*128, sd=10), sinfo)

## apply denoising models
xproc1 <- denoise2(xnoised, lambda=10, method="TVL2.FiniteDifference")
xproc2 <- denoise2(xnoised, lambda=10, method="TVL1.PrimalDual")

## compare
gcol = gray(0:256/256)
x11()
par(mfrow=c(2,2), pty="s")
image(lena128, main="original", col=gcol)
image(xnoised, main="noised", col=gcol)
image(xproc1, main="TVL2.FiniteDifference", col=gcol)
image(xproc2, main="TVL1.PrimalDual", col=gcol)

## End(Not run)
```

`lena128` *lena image at size of* (128×128)

Description

Lena is probably one of the most well-known example in image processing and computer vision. Well, here is a brief introduction on the story of Lena.

Usage

```
data(lena128)
```

Format

matrix of size (128×128)

Source

USC SIPI Image Database

References

Gonzalez, Rafael C. and Woods, Richard E. (2017) *Digital Image Processing* (4th ed.). ISBN 0133356728.

Examples

```
data(lena128)
image(lena128, col=gray((0:100)/100), axes=FALSE, main="lena128")
```

Index

∗Topic **datasets**
    lena128, 6

denoise1, 2
denoise2, 4

lena128, 6

tvR-package, 2