Package ‘cems’

November 13, 2015

Version 0.4
Date 2015-11-8
Author Samuel Gerber <sgerber@uoregon.edu>
Maintainer Samuel Gerber <sgerber@uoregon.edu>

Description
Conditional expectation manifolds are an approach to compute principal curves and surfaces.

Title Conditional Expectation Manifolds
Depends R (>= 2.10)
Imports plotrix, rgl, vegan, stats, graphics, grDevices
License GPL-2
Repository CRAN
NeedsCompilation yes
Date/Publication 2015-11-12 19:09:20

R topics documented:
cem ................................................................. 1
cem.example.arc .................................................. 4
cem.example.sr .................................................... 5
frey_faces .......................................................... 6
swissroll ............................................................. 6

Index 8

cem

---

Conditional Expectation Manifolds

---
Description

This package computes principal surfaces based on the approach described in Gerber et. al. 2009 and Gerber and Whitaker 2011.

Principal surfaces are typically found by minimizing $E[||Y - g(\lambda(Y))||^2]$ over the functions $g : R^m \mapsto R^n$ with $m < n$ and $\lambda : R^m \mapsto R^n$ defined as an orthogonal projection onto $g$.

In Gerber et. al. 2009 the opposite approach is described; fixing $g_\lambda(x) = E[Y|\lambda(Y) = x]$ and minimizing over $\lambda$, i.e. optimizing the conditional expectation manifold (CEM) given $\lambda$. Gerber et. al. 2009 called this approach kernel map manifolds (KMM) since both mappings where defined by kernel regression.

In Gerber and Whitaker 2011 the same formulation is exploited to provide an approach that solves the model selection problem for principal curves. The orthogonal projection distance minimization $E[||Y - g_\lambda(Y)||^2]$ yields principal curves that are saddle points and thus model selection (i.e. bandwidth selection in this implementation) is not possible even with cross-validation. The approach in Gerber and Whitaker 2011 formulates an alternate optimization problem minimizing orthogonality $E[<Y - g_\lambda(Y), \frac{dg(s)}{ds}|_{s=\lambda(Y)}>^2]$ which leads to principal curves at minima.

This package implements the approach in Gerber et. al. 2009 for both formulation, i.e. minimizing projection distance $E[||Y - g_\lambda(Y)||^2]$ or orthogonality $E[<Y - g_\lambda(Y), \frac{dg(s)}{ds}|_{s=\lambda(Y)}>^2]$.

Usage

```r
cem(y, x, knnx=50, sigmaX=1/3, iter =100, npoints = nrow(y), stepX = 0.25, stepBW = 0.1, verbose=1, penalty=0, sigmaAsFactor = T, optimalSigmaX = F, quadratic = F)
cem.optimize(object, iter =100, npoints = nrow(object$y), stepX=1, stepBW=0.1, verbose=1, optimalSigmaX = F)
## S3 method for class 'cem'
predict(object, newdata = object$y, type=c("coordinates", "curvature", ...))
cem.geodesic(object, xs, xe, iter = 100, step = 0.01, verbose=1, ns=100)
```

Arguments

- **y**: $n$-dimensional data to compute conditional expectation manifold for.
- **x**: Initialization for low dimensional mapping $\lambda$. For example an isomap or lle or PCA embedding of $y$.
- **knnX**: Number of nearest neighbors for kernel regression of $g$, i.e. the regression is truncated to only the knnX nearest neighbors.
- **sigmaX**: Initialize bandwidth of $g$ to sigmaX. If sigmaAsFactor is set to true the bandwidth is computed as sigmaX times average knnX nearest neighbor distance.
- **iter**: Number of optimization iterations, i.e. number of gradient descent with line search steps.
**cem**

- **stepX**: Gradient descent step size for optimizing coordinate mapping
- **stepBW**: Gradient descent step size for optimizing bandwidths
- **verbose**: Report gradient descent information. 1 reports iteration number and mean squared projection distances. 2 has additional information on step size and line search.
- **sigmaAsFactor**: Use sigmaX and sigmaY as multipliers of the average nearest neighbor distances in Y and λ(Y) respectively.
- **optimalSigmaX**: If true optimizes sigmaX before every iteration - will not work for MSE minimization, i.e. sigmaX will go to 0 - work well for orthogonal projection and speeds up computation significantly
- **risk**: Which objective function should be minimized. 0 = \(E[|Y - g_\lambda(Y)|^2]\). 1 = \(E[<(g(f(y)) - y, g'(f(y)) \lambda(y)>]^2]\). 2 = 1 but with \(g'(f(y)) > \) ortho normalized. 3 = 2 with \(g(f(y)) - y\) normalized.
- **penalty**: 0 = No penalty, 1 = Deviation from arc length parametrization
- **quadratic**: Use a locally quadratic regression instead of linear for \(g\)
- **nPoints**: Number of points that are sampled for computing gradient descent directions
- **object**: CEM object to do prediction for
- **newdata**: Data to do prediction for. If ncol(newdata) \(= m\) for each point \(x\) in newdata \(g(x)\) is computed. If ncol(newdata) \(= n\) for each point \(y\) in newdata \(\lambda(y)\) is computed.
- **type**: Prediction type: coordinates or curvatures of the manifold model
- **...**: Additional arguments have no effect.
- **xs**: Start point for geodesic
- **xe**: End point for geodesic
- **step**: Step size for optimizing geodesic
- **ns**: Number of segments for discretizing geodesic

**Value**

An object of class "cem".

**Author(s)**

Samuel Gerber

**References**


**See Also**

cem.example.arc  cem.example.sr
Examples

```r
# Noisy half circle example
phi <- runif(100)*pi
arc <- cbind(cos(phi), sin(phi)) * (1+rnorm(100) * 0.1)

pc <- cem(y=arc, x=phi, knn=10, iter=10, optimalSigma=TRUE, verbose=2)

# Predict original data
y <- predict(pc, pc$x)

# Predict new data
xt <- seq(min(pc$x), max(pc$x), length.out=100)
yt <- predict(pc, xt)

# Plot things
arc0 <- cbind(cos(phi), sin(phi))
o0 <- order(phi)

par(mar=c(5,5,4,2))
plot(arc, xlab=expression(y[1]), ylab=expression(y[2]), col = "#00000020",
     pch=19, asp=1, cex.lab=1.5, cex.axis=1.5, cex=2, bty="n")
lines(arc[o0,], lwd=4, col="black", lty=6)
lines(yt$y, col="dodgerblue", lwd=4, lty=1)
```

---

desc.

This function runs the arc example in:


Usage

cem.example.arc(n=150, noise=0.2, risk=2, sigmaX=0.1, stepX=0.001, stepBW=0.01, init = 0, plotEach=1, noiseInit=0.5)

Arguments

- **n**: Sample size.
- **noise**: Amount of normal distributed noise orthogonal to the arc.
- **risk**: Optimization objective
- **sigmaX**: Initial bandwidth for the curve \( g \)
cem.example.sr

stepBW  Stepsize for bandwidth optimization
stepX   Stepsize for coordinate optimization
init    Type of initialization. 0 = ground truth, 1 = random, 2 = y-values of arc (i.e. close to principal component)
plotEach Plot curve after plotEach iterations.
noiseInit add normal distribution noise to initialization.

Author(s)

Samuel Gerber

References


See Also
cem

cem.example.sr  Conditional Expectation Manifold Example on Swissroll

Description

This function runs the swissroll example in:


Usage

cem.example.sr(n =1000, nstd=0.1, init=0, risk=2, stepX=0.1)

Arguments

n  Sample size.
nstd  Amount of normal distributed noise orthogonal to the swissroll.
risk  Optimization objective
init  Type of initialization. 0 = ground truth, 1 = random, 2 = y-values of arc (i.e. close to principal component)
stepX  Optimization step size
Author(s)
Samuel Gerber

References

See Also
cem

grey_faces Frey faces

Description
Set of 1995 face images from a single subject with different facial expression as well as different orientations. (from http://www.cs.nyu.edu/~roweis/data.html)

Author(s)
Samuel Gerber

Examples
```
data("frey_faces")
im <- matrix(faces[1, 560:1], 20, 28)
image(1:nrow(im), 1:ncol(im), im, xlab="", ylab="")
```

swissroll Fourpeaks Function

Description
Swissroll data set.

Author(s)
Samuel Gerber
Examples

```r
library(rgl)
data(swissroll)
# create 1000 samples with standard parameters
d <- swissroll()

# X contains original data
plot3d(d$x)
# Xn contains data with gaussian noise added orthogonally
plot3d(d$xn)

# create 2000 samples with different parameters
# phi - number of revolutions
# nstd - std of normal noise added orthogonally
d <- swissroll(2000, nstd = 0.5, height = 5, phi = 2*pi)
plot3d(d$xn)
```
Index

*Topic **datasets**
  frey_faces, 6
  swissroll, 6
*Topic **nonparametric,models,nonlinear**
  cem, 1
  cem.example.arc, 4
  cem.example.sr, 5
  cem, 1, 5, 6
  cem.example.arc, 3, 4
  cem.example.sr, 3, 5
  cems (cem), 1
  cems-package (cem), 1
  faces (frey_faces), 6
  frey_faces, 6
  predict.cem (cem), 1
  swissroll, 6