Package ‘BiocSklearn’

October 14, 2021

Title interface to python sklearn via Rstudio reticulate

Description This package provides interfaces to selected sklearn elements, and demonstrates fault tolerant use of python modules requiring extensive iteration.

Version 1.14.1

Suggests testthat, restfulSE, HDF5Array, BiocStyle, rmarkdown

Depends R (>= 4.0), reticulate, methods, SummarizedExperiment, knitr

Imports basilisk, Rcpp

License Artistic-2.0

LazyLoad yes

biocViews StatisticalMethod, DimensionReduction, Infrastructure

RoxygenNote 7.1.1

VignetteBuilder knitr

StagedInstall false

SystemRequirements python (>= 2.7), sklearn, numpy, pandas, h5py

Roxygen list(markdown=TRUE)

Encoding UTF-8

git_url https://git.bioconductor.org/packages/BiocSklearn

git_branch RELEASE_3_13

git_last_commit df73d5c

git_last_commit_date 2021-07-28

Date/Publication 2021-10-14

Author Vince Carey [cre, aut]

Maintainer Vince Carey <stvjc@channing.harvard.edu>

R topics documented:

`h5mat` *create a file connection to HDF5 matrix*

Description

create a file connection to HDF5 matrix

Usage

```
h5mat(infile, mode = "r", ...)
```

Arguments

`infile`	a pathname to an HDF5 file
`mode`	character(1) defaults to "r", see py_help for h5py.File
`...`	unused

Value

instance of (S3) h5py._hl.files.File

Note

The result of this function must be used with basiliskRun with the env argument set to bsklenv, or there is a risk of inconsistent python modules being invoked. This should only be used with the persistent environment discipline of basilisk.

Examples

```
if (interactive()) {   # not clear why
fn = system.file("ban_6_17/assays.h5", package="BiocSklearn")
proc = basilisk::basiliskStart(BiocSklearn:::bsklenv)
basilisk::basiliskRun(proc, function(infile, mode="r") {
 h5py = reticulate::import("h5py")
 hh = h5py$File( infile, mode=mode )
 cat("File reference:\n ")
 print(hh)
 cat("File attributes in python:\n ")
 print(head(names(hh)))
 cat("File keys in python:\n ")
 print(hh$keys())
 cat("HDF5 dataset in python:\n ")
 print(hh['assay001'])
}, infile=fn, mode="r")
basilisk::basiliskStop(proc)
}
```

`H5matref` *obtain an HDF5 dataset reference suitable for handling as numpy matrix*

Description

obtain an HDF5 dataset reference suitable for handling as numpy matrix

Usage

```
H5matref(filename, dsname = "assay001")
```

Arguments

`filename`	a pathname to an HDF5 file
`dsname`	internal name of HDF5 matrix to use, defaults to 'assay001'

Value

instance of (S3) "h5py._hl.dataset.Dataset"

Note

This should only be used with persistent environment discipline of basilisk. Additional support is planned in Bioc 3.12.

Examples

```
## Not run:
fn = system.file("ban_6_17/assays.h5", package="BiocSklearn")
ban = H5matref(fn)
ban
proc = basilisk::basiliskStart(BiocSklearn:::bsklenv)
basilisk::basiliskRun(proc, function() {
 np = import("numpy", convert=FALSE) # ensure
 print(ban$shape)
 print(np$take(ban, 0:3, 0L))
 fullpca = skPCA(ban)
 dim(getTransformed(fullpca))
 ta = np$take
 })
basilisk::basiliskStop(proc)

## End(Not run)
# project samples
## Not run:   # on celaya2 this code throws errors, and
#  I have seen
# .../lib/python2.7/site-packages/sklearn/decomposition/incremental_pca.py:271: RuntimeWarning: Mean of empty sl
#   explained_variance[self.n_components_:].mean()
# .../lib/python2.7/site-packages/numpy/core/_methods.py:85: RuntimeWarning: invalid value encountered in double_
#   ret = ret.dtype.type(ret / rcount)
ta(ban, 0:20, 0L)$shape
st = skPartialPCA_step(ta(ban, 0:20, 0L))
st = skPartialPCA_step(ta(ban, 21:40, 0L), obj=st)
st = skPartialPCA_step(ta(ban, 41:63, 0L), obj=st)
oo = st$transform(ban)
dim(oo)
cor(oo[,1:4], getTransformed(fullpca)[,1:4])

## End(Not run) # so blocking this part of example for now
```

SkDecomp *constructor for SkDecomp*

Description

constructor for SkDecomp

Usage

```
SkDecomp(transform, method)
```

Arguments

transform	typically a numerical matrix representing a projection of the input
method	character(1) arbitrary tag describing the decomposition

SkDecomp-class *container for sklearn objects and transforms*

Description

container for sklearn objects and transforms

Usage

```
## S4 method for signature 'SkDecomp'
getTransformed(x)
```

Arguments

`x`	instance of SkDecomp

Value

the getTransformed method returns a matrix

Slots

`transform` stored as R matrix

`method` string identifying method

Note

In Bioc 3.11, the `object` slot is removed. This is a consequence of adoption of basilisk discipline for acquiring and using python resources, which greatly increases reliability, at the expense of added complication in handling python objects interactively in R. We are working on restoring this functionality but it will take time.

`skIncrPartialPCA` *use basilisk discipline to perform partial (n_components) incremental (chunk.size) PCA with scikit.decomposition*

Description

use basilisk discipline to perform partial (n_components) incremental (chunk.size) PCA with scikit.decomposition

Usage

```
skIncrPartialPCA(mat, n_components, chunk.size = 10)
```

Arguments

`mat`	a matrix
`n_components`	integer(1) number of PCs to compute
`chunk.size`	integer(1) number of rows to use each step

Note

A good source for capabilities and examples is at the sklearn doc site.

Examples

```
lk = skIncrPartialPCA(iris[,1:4], n_components=3L)
lk
head(getTransformed(lk))
```

`skIncrPCA` *use sklearn IncrementalPCA procedure*

Description

use sklearn IncrementalPCA procedure

Usage

```
skIncrPCA(mat, n_components = 2L, batch_size = 5L, ...)
```

Arguments

`mat`	a matrix – can be R matrix or numpy.ndarray, if the latter it must be set up in a basilisk persistent environment, and that is not currently demonstrated for this package.
`n_components`	number of PCA to retrieve
`batch_size`	number of records to use at each iteration
`...`	passed to python IncrementalPCA

Value

matrix with rotation

Examples

```
dem = skIncrPCA(iris[,1:4], batch_size=25L)
## Not run:
# this is the unsupported way
irloc = system.file("csv/iris.csv", package="BiocSklearn")
irismat = SklearnEls()$np$genfromtxt(irloc, delimiter=',')
ski = skIncrPCA(irismat)
ski25 = skIncrPCA(irismat, batch_size=25L) # non-default
getTransformed(ski)[1:3,]
getTransformed(ski25)[1:3,]

## End(Not run) # end dontrun
dem = skIncrPCA(iris[,1:4], batch_size=25L, n_components=2L)
dem
```

`skIncrPCA_h5` *demo of HDF5 processing with incremental PCA/batch_size/fit_transform*

Description

demo of HDF5 processing with incremental PCA/batch_size/fit_transform

Usage

```
skIncrPCA_h5(fn, dsname = "assay001", n_components, chunk.size = 10L)
```

Arguments

`fn`	character(1) path to HDF5 file
`dsname`	character(1) name of dataset within HDF5 file, assumed to be 2-dimensional array
`n_components`	numeric(1) passed to IncrementalPCA
`chunk.size`	numeric(1) passed to IncrementalPCA as batch_size

Note

Here we use IncrementalPCA$fit_transform and let python take care of chunk retrieval. `skIncrPartialPCA` acquires chunks from R matrix and uses IncrementalPCA$partial_fit.

Examples

```
if (interactive()) {
 fn = system.file("hdf5/irmatt.h5", package="BiocSklearn") # 'transposed' relative to R iris
 dem = skIncrPCA_h5(fn, n_components=3L, dsname="tquants")
 dem
 head(getTransformed(dem))
}
```

`skIncrPPCA` *optionally fault tolerant incremental partial PCA for projection of samples from SummarizedExperiment*

Description

optionally fault tolerant incremental partial PCA for projection of samples from SummarizedExperiment

Usage

```
skIncrPPCA(
  se,
  chunksize,
  n_components,
  assayind = 1,
  picklePath = "./skIdump.pkl",
  matTx = force,
  ...
)
```

Arguments

`se`	instance of SummarizedExperiment
`chunksize`	integer number of samples per step
`n_components`	integer number of PCs to compute
`assayind`	not used, assumed set to 1
`picklePath`	if non-null, incremental results saved here via sklearn.externals.joblib.dump, for each chunk. If NULL, no saving of incremental results.
`matTx`	a function defaulting to force() that accepts a matrix and returns a matrix with identical dimensions, e.g., `function(x) log(x+1)`
`...`	not used

Value

python instance of `sklearn.decomposition.incremental_pca.IncrementalPCA`

Note

Will treat samples as records and all features (rows) as attributes, projecting. to an `n_components`-dimensional space. Method will acquire chunk of assay data and transpose before computing PCA contributions. In case of crash, restore from `picklePath` using `SklearnEls()$joblib$load` after loading reticulate. You can use the `n_samples_seen_` component of the restored python reference to determine where to restart. You can manage resumption using `skPartialPCA_step`.

Examples

```
## Not run:
# demo SE made with TENxGenomics:
# mm = matrixSummarizedExperiment(h5path, 1:27998, 1:750)
# saveHDF5SummarizedExperiment(mm, "tenx_750")
#
if (requireNamespace("HDF5Array")) {
  se750 = HDF5Array::loadHDF5SummarizedExperiment(
     system.file("hdf5/tenx_750", package="BiocSklearn"))
  lit = skIncrPPCA(se750[, 1:50], chunksize=5, n_components=4)
  round(cor(pypc <- lit$transform(dat <- t(as.matrix(assay(se750[,1:50]))))),3)
  rpc = prcomp(dat)
  round(cor(rpc$x[,1:4], pypc), 3)
}

## End(Not run) # this has to be made basilisk-compliant
```

---

skKMeans *interface to sklearn.cluster.KMeans using basilisk discipline*

Description

interface to sklearn.cluster.KMeans using basilisk discipline

Usage

```
skKMeans(mat, ...)
```

Arguments

mat	a matrix-like datum or reference to such
...	arguments to sklearn.cluster.KMeans

Value

a list with cluster assignments (integers starting with zero) and asserted cluster centers.

Note

You can use `py_help(SklearnEls()$skcl$KMeans)` to get python documentation on parameters and return structure. This is a demonstrative interface to the resources of sklearn.cluster. In this particular interface, we are using sklearn.cluster.k_means_.KMeans. There are many other possibilities in sklearn.cluster: _dbscan_inner, *feature_agglomeration,* hierarchical, *k_means,* k_means_elkan, affinity_propagation*, bicluster, birch, dbscan*, hierarchical, k_means*, mean_shift*, setup, spectral.

Examples

```
## Not run:
# This blocked example shows a risky approach evading basilisk discipline and is
# to be used at your own risk.
# start with numpy array reference as data
irloc = system.file("csv/iris.csv", package="BiocSklearn")
skels = SklearnEls()
irismat = skels$np$genfromtxt(irloc, delimiter=',')
ans = skKMeans(irismat, n_clusters=2L)
names(ans) # names of available result components
table(iris$Species, ans$labels_)
# now use an HDF5 reference
irh5 = system.file("hdf5/irmat.h5", package="BiocSklearn")
fref = skels$h5py$File(irh5)
ds = fref$`__getitem__`("quants") # thanks Samuela Pollack!
```

```
ans2 = skKMeans(skels$np$array(ds)$T, n_clusters=2L) # HDF5 matrix is transposed relative to python array layout!
table(ans$labels_, ans2$labels_)
ans3 = skKMeans(skels$np$array(ds)$T,
   n_clusters=8L, max_iter=200L,
   algorithm="full", random_state=20L)

## End(Not run)
dem = skKMeans(iris[,1:4], n_clusters=3L, max_iter=100L, algorithm="full",
   random_state=20L)
str(dem)
tab = table(iris$Species, dem$labels)
tab
plot(iris[,1], iris[,3], col=as.numeric(factor(iris$Species)))
points(dem$centers[,1], dem$centers[,3], pch=19, col=apply(tab,2,which.max))
```

`SklearnEls` *mediate access to python modules from sklearn.decomposition*

Description

mediate access to python modules from sklearn.decomposition

Usage

```
SklearnEls()
```

Value

list of (S3) "python.builtin.module"

Note

Deprecated. Previously returned a list with elements np (numpy), pd (pandas), h5py (h5py), skd (sklearn.decomposition), joblib (sklearn.externals.joblib), each referring to python modules. This was done directly with reticulate, but basilisk package discipline is more reliable.

Examples

```
## Not run:
els = SklearnEls()
names(els$skd) # slow at first
# try py_help(els$skd$PCA) # etc.

## End(Not run)
```

`skPartialPCA_step` *take a step in sklearn IncrementalPCA partial fit procedure*

Description

take a step in sklearn IncrementalPCA partial fit procedure

Usage

```
skPartialPCA_step(mat, n_components, obj)
```

Arguments

`mat`	a matrix – can be R matrix or numpy.ndarray
`n_components`	number of PCA to retrieve
`obj`	sklearn.decomposition.IncrementalPCA instance

Value

trained IncrementalPCA reference, to which 'transform' method can be applied to obtain projection for any compliant input

Note

if obj is missing, the process is initialized with the matrix provided

Examples

```
## Not run:
# these steps are not basilisk-compliant, you need to acquire references
irloc = system.file("csv/iris.csv", package="BiocSklearn")
irismat = SklearnEls()$np$genfromtxt(irloc, delimiter=',')
ta = SklearnEls()$np$take
ipc = skPartialPCA_step(ta(irismat,0:49,0L))
ipc = skPartialPCA_step(ta(irismat,50:99,0L), obj=ipc)
ipc = skPartialPCA_step(ta(irismat,100:149,0L), obj=ipc)
head(names(ipc))
ipc$transform(ta(irismat,0:5,0L))
fullproj = ipc$transform(irismat)
fullpc = prcomp(data.matrix(iris[,1:4]))$x
round(cor(fullpc,fullproj),3)

## End(Not run)
```

skPCA *use sklearn PCA procedure*

Description

use sklearn PCA procedure

Usage

```
skPCA(mat, ...)
```

Arguments

`mat`	a matrix – can be R matrix or numpy.ndarray
`...`	additional parameters passed to sklearn.decomposition.PCA, for additional information use `py_help()` on a `reticulate-imported sklearn.decomposition.PCA instance.`

Value

matrix with rotation

Note

If no additional arguments are passed, all defaults are used.

Examples

```
#irloc = system.file("csv/iris.csv", package="BiocSklearn")
#irismat = SklearnEls()$np$genfromtxt(irloc, delimiter=',')
#skpi = skPCA(irismat)
#getTransformed(skpi)[1:5,]
chk = skPCA(data.matrix(iris[,1:4]))
chk
head(getTransformed(chk))
head(prcomp(data.matrix(iris[,1:4]))$x)
```

Index