Package 'VorVQ'

October 13, 2020

Type Package

Title Geometry of Voronoi Tessellations of a Vector Quantizer

Version 1.0

Date 2019-05-09

Author Josh Taylor

Maintainer Josh Taylor <josh@somdisco.com>

Description Tools for describing the geometry of the first and second order Voronoi tessellation induced by the prototypes of a vector quantizer.

License MIT + file LICENSE

Depends Rcpp (>= 1.0.0)

Imports Rcpp (>= 1.0.0),
RcppArmadillo,
RcppParallel,
Rdpack,
rcdd,
igraph

LinkingTo Rcpp, RcppArmadillo, RcppParallel

RoxygenNote 7.1.1

RdMacros Rdpack

LazyData true

R topics documented:

`as_list` *Convert an VOR object to a list*

Description

This method extracts all fields of an VOR object and places their stored values into the fields of an R list, with list field names matching the VOR object field names.

Usage

```
VORobj$as_list()
```

Value

A list

---

calc_all *Calc all Voronoi quantities*

Description

This wrapper method computes all Voronoi quantities exposed by the VOR class, in the following order:

- `calc_DADJ`
- `calc_vor2_centers`
- `calc_vor1_Dikin`
- `calc_vor2_Dikin`
- `calc_vor1_MVIE`
- `calc_vor2_MVIE`

Usage

```
VORobj$calc_all()
```

---

calc_DADJ *Calculate the Delaunay graph adjacency*

Description

The Delaunay graph dual of a first-order Voronoi tessellation has edges between prototype vertices whose corresponding Voronoi cells intersect (share a face). This method computes the (binary) adjacency matrix of this graph using the method Agrell (referenced below), which involves solving a linear program for each unique pair of prototypes in the tessellation. To speed up the computation, the CADJ and Gabriel adjacency matrices (set during `initialize_VOR`) are used to "seed" the Delaunay adjacency (as proper sub-graphs of the Delaunay graph, the existence of an edge between prototypes `i` and `j` in either necessarily indicates a corresponding edge in DADJ). Note: a Delaunay adjacency is required for all ellipsoidal calculations in VorVQ. As the many thousand LPs required for full DADJ calculation can be quite large it is recommended to perform this calculation in parallel, which can be set in set_params (default is `parallel = TRUE`).

Usage

```
VORobj$calc_DADJ
```

Value

None, the field `DADJ` is set internally.

References

Agrell E (1993). "A method for examining vector quantizer structures." In *Proceedings. IEEE International Symposium on Information Theory*, 394–394. IEEE.

`calc_GADJ` *Calculate the Gabriel graph adjacency*

Description

The Gabriel graph is a proximity graph that is a known sub-graph of the Delaunay triangulation induced by the Voronoi tessellation of the prototypes in `W`. The Gabriel ADJacency matrix helps speed up the computation of the full Delaunay graph and is calculated automatically during `initialize_VOR` via an internal call to `calc_GADJ`. This method should not need to be called directly by a user but it exposed for completeness.

Usage

```
VORobj$calc_GADJ()
```

Details

Field `GADJ` will be overwritten internally.

Value

None

References

Gabriel KR, Sokal RR (1969). "A New Statistical Approach to Geographic Variation Analysis." *Systematic Zoology*, **18**(3), 259–278. ISSN 00397989, http://www.jstor.org/stable/2412323.

`calc_vor1_centers` *Calculate the Chebyshev centers of first-order Voronoi cells*

Description

The Chebyshev center of the polytope definition (from `get_vor1_polytope`) for each **active** first-order Voronoi cell (as specified in `vor1_active`) is computed. If set, these centers are used as starting interior points for MVIE and Dikin ellipsoid calculations. Clear any previously computed centers via `clear_vor1_centers`.

Usage

```
VORobj$calc_vor1_centers
```

Value

None, the field `vor1_centers` is set internally.

calc_vor1_Dikin *Compute the Dikin ellipsoid for first-order Voronoi cells*

Description

The Dikin ellipsoid is a locally inscribed geometric approximator of a polytope. These ellipsoidal approximators are computed for each active first-order Voronoi cell at the points specified in `vor1_centers` (if these have been set); otherwise the active prototypes in `W` are used.

Usage

```
VORobj$calc_vor1_Dikin()
```

Details

Details of the Dikin ellipsoid can be found in the help of Dikin_Ellipsoid.

Value

None, the field `vor1_MVIE_E`, which is a cube whose slices contain the `vor1_active` Dikin ellipsoid rotations, is stored internally.

References

Dikin I (1967). "Iterative solution of problems of linear and quadratic programming." In *Doklady Akademii Nauk*, volume 174(4), 747–748. Russian Academy of Sciences. Boyd S, Boyd SP, Vandenberghe L (2004). *Convex optimization*. Cambridge university press.

calc_vor1_MVIE *Compute the MVIE for first-order Voronoi cells*

Description

The **M**aximum **V**olume **I**nscribed **E**llipsoid inside the polytope defined by each active first-order Voronoi cell is computed and stored, according to the method of Zhang & Gao. The starting points for the routine are taken from `vor1_centers`, if it has been set (via either `calc_vor1_centers` or `set_vor1_centers`); otherwise the prototypes stored in `W` are used. The parameters controlling the routine can be viewed or changed via `get/set_params`. It is recommended to allow parallel computation for MVIE calculation (i.e., `set_params(parallel = TRUE)`.

Usage

```
VORobj$calc_vor1_MVIE()
```

Details

Details of the MVIE routine can be found in the help of max_vol_inscr_ell.

Value

None, the following fields store components of the MVIE:

`vor1_MVIE_c` matrix of ellipsoid centers, rows correspond to entries in `vor1_active`

`vor1_MVIE_E` cube of ellipsoid rotation matrices, slices correspond to entries in `vor1_active`

`vor1_MVIE_logdet` vector of the log-determinant of the ellipsoid rotations stored in the slices of `vor1_MVIE_E`

`vor1_MVIE_status` vector of the return flags from calling the MVIE routinefor each `vor1_active` cell

`vor1_MVIE_volratio` vector of the proportional volume of each `vor1_active` MVIE (should sum to unity)

References

Zhang Y, Gao L (2003). "On Numerical Solution of the Maximum Volume Ellipsoid Problem." *SIAM Journal on Optimization*, **14**(1), 53-76. doi: 10.1137/S1052623401397230, https://doi.org/10.1137/S10526234013
https://doi.org/10.1137/S1052623401397230.

---

calc_vor2_centers *Calculate the Chebyshev centers of second-order Voronoi cells*

Description

The Chebyshev center of the polytope definition (from `get_vor2_polytope`) for each **active** second-order Voronoi cell (as specified in `vor2_active`) is computed. These centers are used as starting interior points for MVIE and Dikin ellipsoid calculations. Clear any previously computed centers via `clear_vor2_centers`.

Usage

```
VORobj$calc_vor2_centers
```

Value

None, the field `vor2_centers` is set internally.

---

calc_vor2_Dikin *Compute the Dikin ellipsoid for second-order Voronoi cells*

Description

The Dikin ellipsoid is a locally inscribed geometric approximator of a polytope. These ellipsoidal approximators are computed for each active second-order Voronoi cell at the points specified in `vor2_centers`

Usage

```
VORobj$calc_vor2_Dikin()
```

Details

Details of the Dikin ellipsoid can be found in the help of Dikin_Ellipsoid.

Value

None, the field `vor2_MVIE_E`, which is a cube whose slices contain the `vor2_active` Dikin ellipsoid rotations, is stored internally.

References

Dikin I (1967). "Iterative solution of problems of linear and quadratic programming." In *Doklady Akademii Nauk*, volume 174(4), 747–748. Russian Academy of Sciences. Boyd S, Boyd SP, Vandenberghe L (2004). *Convex optimization*. Cambridge university press.

`calc_vor2_MVIE` *Compute the MVIE for second-order Voronoi cells*

Description

The **M**aximum **V**olume **I**nscribed **E**llipsoid inside the polytope defined by each active second-order Voronoi cell is computed and stored, according to the method of Zhang & Gao. The starting points for the routine are taken from `vor2_centers`, which must be populated prior to calling this method (via either `calc_vor2_centers` or `set_vor2_centers`). The parameters controlling the routine can be viewed or changed via `get/set_params`. It is recommended to allow parallel computation for MVIE calculation (i.e., `set_params(parallel = TRUE)`.

Usage

```
VORobj$calc_vor2_MVIE()
```

Details

Details of the MVIE routine can be found in the help of max_vol_inscr_ell.

Value

None, the following fields store components of the MVIE:

`vor2_MVIE_c` matrix of ellipsoid centers, rows correspond to rows of `vor2_active`

`vor2_MVIE_E` cube of ellipsoid rotation matrices, slices correspond to rows of `vor2_active`

`vor2_MVIE_logdet` vector of the log-determinant of the ellipsoid rotations stored in the slices of `vor2_MVIE_E`

`vor2_MVIE_status` vector of the return flags from calling the MVIE routinefor each `vor2_active` cell

`vor2_MVIE_volratio` vector of the proportional volume of each `vor2_active` MVIE (should sum to unity)

References

Zhang Y, Gao L (2003). "On Numerical Solution of the Maximum Volume Ellipsoid Problem." *SIAM Journal on Optimization*, **14**(1), 53-76. doi: 10.1137/S1052623401397230, https://doi.org/10.1137/S10526234013
https://doi.org/10.1137/S1052623401397230.

clear_vor1_centers *Clear the centers of first-order Voronoi cells*

Description

This method clears any values stored in the field `vor1_centers` that were previously populated by either `calc_vor1_centers` or `set_vor1_centers`.

Usage

```
VORobj$clear_vor1_centers
```

Value

None, the field `vor1_centers` is reset to a matrix with `nrows = ncols = 0`.

clear_vor2_centers *Clear the centers of second-order Voronoi cells*

Description

This method clears any values stored in the field `vor2_centers` that were previously populated by either `calc_vor2_centers` or `set_vor2_centers`.

Usage

```
VORobj$clear_vor2_centers
```

Value

None, the field `vor2_centers` is reset to a matrix with `nrows = ncols = 0`.

CMIX9 *A Synthethic 2-D Gaussian Mixture*

Description

Table 1 of (Chacón 2009) compiles twelve different two-dimensional synthetic Gaussian mixtures which have been exercised extensively for the task of density estimation. For demonstration of Voronoi calculations and visualization in lower dimension we provide Mixture 9 in `VorVQ`.

Usage

```
CMIX9
```

Format

These data are stored in a list variable named `CMIX9` with components:

X data matrix with 5000 rows and 2 columns

W matrix of 100 learned prototype vectors (in rows, learned from a 10x10 SOM)

CADJ The CADJ matrix resulting from a recall of the data in `X` using the prototypes in `W`. `CADJ` is ordered such that its `(i,j)` element contains the number of data vectors in `X` whose BMU1 = the prototype vector in the `i`-th row and BMU2 = the prototype in the `j`-th row of `W`

References

Chacón JE (2009). "Data-driven choice of the smoothing parametrization for kernel density estimators." *Canadian Journal of Statistics*, **37**(2), 249–265.

`Delaunay_ADJ` *Build the Delaunay Adjacency Matrix of a set of prototypes*

Description

Build the Delaunay Adjacency Matrix of a set of prototypes

Usage

```
Delaunay_ADJ(W, lb, ub, parallel = TRUE)
```

Arguments

`W`	prototype matrix, with prototype vectors in rows
`lb`	a vector of the global (data range) lower bounds, by dimension
`ub`	a vector of the global (data range) upper bounds, by dimension
`parallel`	whether to process in parallel. Default = TRUE.

Value

A (binary, symmetric) adjacency matrix, nrows = ncols = nrow(W), giving the Delaunay adjacencies between all prototypes in W.
The global lower and upper bounds (xlb and xub) are needed to ensure the polyope definition is bounded. Otherwise, the LP we need to solve here could be unbounded, which produces incorrect results.

References

Delaunay B, others (1934). "Sur la sphere vide." *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, **7**(793-800), 1–2.

Agrell E (1993). "A method for examining vector quantizer structures." In *Proceedings. IEEE International Symposium on Information Theory*, 394–394. IEEE.

Dikin_ell *Find Dikin ellipsoids inside a polytope*

Description

Returns a matrix E such that the ellipsoid $(y - c)'E^-1(y - c) <= 1$ is inscribed in the polytope defined by Ax <= b, with curvature influenced by the local geometry of the constraints at c

Usage

```
Dikin_ell(A, b, c)
```

Arguments

A	LHS of polytope constraints
b	RHS of polytope constraints
c	interior point where Dikin ellipse is centered

Value

the rotation matrix of the Dikin ellipsoid centered at c.

References

Dikin I (1967). "Iterative solution of problems of linear and quadratic programming." In *Doklady Akademii Nauk*, volume 174(4), 747–748. Russian Academy of Sciences. Boyd S, Boyd SP, Vandenberghe L (2004). *Convex optimization*. Cambridge university press.

Gabriel_ADJ *Compute the Gabriel graph adjacency*

Description

Compute the Gabriel graph adjacency

Usage

```
Gabriel_ADJ(W, parallel = TRUE)
```

Arguments

W	the prototype matrix, prototype vectors in rows
parallel	boolean, whether to compute in parallel. Default = TRUE.

Details

The Gabriel graph is a proximity graph of points that is known to be a sub-graph of the Delaunay triangulation of the same set of points.

Value

a (binary, symmetric) adjacency matrix of dimension `nrow(W) x nrow(W)`. Element `(i,j) = 1` IFF the prototype vectors `W[i,]` and `W[j,]` are Gabriel adjacent. Otherwise, entries are 0.

References

Gabriel KR, Sokal RR (1969). "A New Statistical Approach to Geographic Variation Analysis." *Systematic Zoology*, **18**(3), 259–278. ISSN 00397989, http://www.jstor.org/stable/2412323.

`get_params` *Get parameters for Voronoi calculations*

Description

Several methods require parameters whose values can be accessed via this method. See set_params for a description of each.

Usage

```
VORobj$get_params()
```

Value

A list with named components equal to the parameters.

`get_vor1_centers` *Get the centers of first-order Voronoi cells*

Description

If `calc_vor1_centers` or `set_vor1_centers` has been set, this method returns the centers calculated or set by these methods. Otherwise, the `vor1_active` prototypes in `W` are returned.

Usage

```
VORobj$get_vor1_centers
```

get_vor1_polytope *Get the definition of a first-order Voronoi cell*

Description

The definition of a first-order Voronoi cell naturally induces a half-plane representation of each cell, of the form $Ax <= b$. This method will return this half-plane definition for a particular cell. By convention, the dimension-wise lower and upper bounds stored in `lb` and `ub` are appended to the system to ensure the resulting polytope is closed. Since constraints arising from any non-Delaunay adjacent prototypes (as defined in `DADJ`) are redundant in the above system, these are not included in the returned polytope definition (making it non-redundant, as small as possible).

Usage

```
VORobj$get_vor1_polytope(iidx)
```

Arguments

`iidx`	an integer identifying which first-order Voronoi cell's definition is returned. It should correspond to the row index of the generator in `W`.

Value

A list with components:

A The LHS coefficient matrix of the linear inequality system

b The RHS upper bounds of the linear system, length = `nrow(A)`

cid A vector of indices (length = `nrow(A)`) indicating which prototype (generator) induced the half-plane constraint stored in the corresponding row of `A` and `b`. Constraints involving the global bounds have `cid` = `nrow(W)+1` by convention.

References

Agrell E (1993). "A method for examining vector quantizer structures." In *Proceedings. IEEE International Symposium on Information Theory*, 394–394. IEEE.

get_vor2_centers *Get the centers of second-order Voronoi cells*

Description

Returns the second-order centers that were set when calling either `calc_vor2_centers` or `set_vor2_centers`.

Usage

```
VORobj$get_vor2_centers
```

`get_vor2_polytope` *Get the definition of a second-order Voronoi cell*

Description

The definition of a second-order Voronoi cell naturally induces a half-plane representation of each cell, of the form $Ax <= b$. This method will return this half-plane definition for a particular cell. By convention, the dimension-wise lower and upper bounds stored in `lb` and `ub` are appended to the system to ensure the resulting polytope is closed. Since constraints arising from any non-Delaunay adjacent prototypes (as defined in `DADJ`) are redundant in the above system, these are not included in the returned polytope definition (making it non-redundant, as small as possible).

Usage

```
VORobj$get_vor2_polytope(iidx, jidx)
```

Arguments

`iidx`

`jidx`

Details

`iidx` and `jidx` are integer indices (to the generators specified in the rows of `W`) which, together, define the second-order Voronoi cell `iidx-jidx`.

Value

A list with components:

A The LHS coefficient matrix of the linear inequality system

b The RHS upper bounds of the linear system, length = `nrow(A)`

cid A vector of indices (length = `nrow(A)`) indicating which prototype (generator) induced the half-plane constraint stored in the corresponding row of `A` and `b`. Constraints involving the global bounds have `cid` = `nrow(W)+1` by convention.

References

Agrell E (1993). "A method for examining vector quantizer structures." In *Proceedings. IEEE International Symposium on Information Theory*, 394–394. IEEE.

initialize_VOR *Initialize a VOR object*

Description

Sets up a VOR object to compute various quantities related to the Voronoi tessellation generated by a vector quantizer.

Usage

```
VORobj$initialize_VOR(W, CADJ)
```

Arguments

W	matrix of prototypes (codebook vectors) in rows
CADJ	the CADJ adjacency matrix produced during recall of a vector quantizer.

Details

This is a wrapper function to perform most steps necessary to setup a VOR object for further calculation, setting internal variables:

- W, nW
- d
- lb, ub
- CADJ
- vor1_active, vor2_active
- GADJ

Value

None

is_Delaunay_nhb *Determine if two prototypes are Delaunay neighbors*

Description

Determine if two prototypes are Delaunay neighbors

Usage

```
is_Delaunay_nhb(W, iidx, jidx, lb, ub, verbose = TRUE)
```

Arguments

W	prototype matrix, with prototype vectors in rows
iidx	the first index of the neighbor pair to test
jidx	the second index of the neighbor pair to test
lb	a vector of the global (data range) lower bounds, by dimension
ub	a vector of the global (data range) upper bounds, by dimension
verbose	boolean, whether to display the LP solver status

Details

iidj and jidx should be valid row indices of W indexed from 1 (not 0).
The global lower and upper bounds (xlb and xub) are needed to ensure the polyope definition is bounded. Otherwise, the LP we need to solve here could be unbounded, which produces incorrect results.

Value

1 = prototypes iidx and jidx are neighbors, 0 = they are not

References

Delaunay B, others (1934). "Sur la sphere vide." *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, **7**(793-800), 1–2.

Agrell E (1993). "A method for examining vector quantizer structures." In *Proceedings. IEEE International Symposium on Information Theory*, 394–394. IEEE.

---

is_interior_point *Check if point is interior*

Description

For a feasible set of linear equations Ax<=b that define a bounded polytope P and a point x0, x0 is interior IFF b - Ax > 0.

Usage

```
is_interior_point(A, b, x)
```

Arguments

A	the LHS matrix of polytope constraints
b	the RHS vector of polytope constraints
x	the query point

Value

boolean = T if the query point satisfies the above condition

load — *Load an existing VOR object*

Description

VOR objects previously written to disk via the save method can be re-loaded into a new R environment with this function. All fields of the internal C++ class will be populated, and all methods can be called on the loaded VOR object.

Usage

```
VORobj$load(vorfile)
```

Arguments

vorfile	a string indicating the file path and name of the saved VOR object.

Details

Because the .vor file is in .rds format it can, technically, be loaded directly into an R environment as a list via readRDS. This can be useful for spot checking the contents of a saved VOR object, but does not allow use of any of its methods (or visualizations). The load methods allows for proper restoration of a previously saved VOR

Value

None, the VOR object is loaded

load_list — *Populate a VOR object from a list*

Description

This method populates all fields of a VOR object from the fields of an R list object. The list must have field names which exactly match the VOR field names.

Usage

```
VORobj$load_list(VORList)
```

Arguments

VORList	a VOR object converted to a list, e.g., with as_list

Value

None

`max_vol_inscr_ell` *Max volume inscribed ellipsoid of polytope*

Description

Returns the matrix E and vector c such that the ellipsoid $(y - c)'E^-1(y - c) <= 1$ is inscribed in the polytope defined by Ax <= b with maximum volume. Code taken from Yin Zhang's MVE routine.

Usage

```
max_vol_inscr_ell(
  A,
  b,
  c0,
  maxiter = 100L,
  tol = 1e-04,
  fix_c0 = FALSE,
  verbose = FALSE
)
```

Arguments

`A`	matrix of LHS constraints in H-representation of polytope
`b`	vector of RHS constraints in H-representation of polytope
`c0`	a point inside the polytope (not on the boundary)
`maxiter`	integer number of iterations to perform, default = 100
`tol`	tolerance for Zhang's MVE code, see paper, default = 1e-4
`fix_c0`	boolean, whether to fix the given center at c0 during iteration, default = F
`verbose`	boolean, whether to display the solver status at each iteration, default = F

Details

maxiter and tol are control (convergence) parameters for Yin Zhang's MVE routine.

The polytope defined by A & b must be bounded.

status = 0 indicates successful convergence; = -1 means maxiter was reached before convergence.

Value

a list with components

- Ea matrix defining the rotation and scale of the maximum volume ellipsoid
- ca vector defining the center of the maximum volume ellipsoid
- statusa flag returned from Zhang's solver code. See details.

References

Zhang Y, Gao L (2003). "On Numerical Solution of the Maximum Volume Ellipsoid Problem." *SIAM Journal on Optimization*, **14**(1), 53-76. doi: 10.1137/S1052623401397230, https://doi.org/10.1137/S10526234013
https://doi.org/10.1137/S1052623401397230.

`new` *Create an empty VOR object*

Description

Create an empty VOR object

Usage

```
VOR$new()
```

Value

an empty VOR object templated for initialization (via initialize_VOR).

`polytope_Chebyshev_center` *Chebyshev center of a polytope*

Description

For a feasible set of linear equations Ax<=b that define a bounded polytope P, the Chebyshev center is the center the largest inscribed ball contained in P. Polytope definition should be given as Ax <= b.

Usage

```
polytope_Chebyshev_center(A, b)
```

Arguments

`A`	the LHS matrix of polytope constraints
`b`	the RHS vector of polytope constraints

Value

a vector (of the same dimension as A) giving the interior point

polytope_chull *Compute the convex hull of vertices of a polytope*

Description

Polytope definition should be given in the H-representation Ax <= b. This function uses Komei Fukuda's cdd library to convert it to a V-representation.

Usage

```
polytope_chull(A, b)
```

Arguments

A	the LHS constraint matrix
b	the RHS constraint vector

Value

a list with components:

1. V a matrix with vertex points in its rows
2. Vpath a 2 column edge list matrix whose rows specify vertex indices (index to rows of V) that are connected in the convex hull path. Note that this path has its redundancies removed (i.e., a connection between vertex 1 & 2 is given only by a row (1,2), not by (2,1))

save *Save a VOR object*

Description

All fields in a VOR object can be saved to disk with this function, which allows them to be re-loaded into a new R environment at a later time (for analysis, or possibly extended training).

Usage

```
VORobj$save(vorfile)
```

Arguments

vorfile	a string indicating the file path and name in which to save the VOR object. This must end in extension ".vor", otherwise an error is returned.

Details

The VOR object is saved to disk as an R list, with each field occupying a corresponding field of the list. The file is saved in .rds format (can check its details with infoRDS).

Saved VORs can be re-loaded with load

Value

None, the VOR object is saved to disk

set_bounds *Set the bounds of the tessellated region*

Description

To ensure that all Voronoi cells are closed in R^d dimension-wise lower and upper bounds of the entire tessellated region must be specified. By default, the upper and lower bounds for each dimension are set = +/- 5 to change these defaults to a user specified range, which must span the entire range of W in each dimension. This will be checked internally. Note that changing the global bounds can alter the Delaunay edges between prototypes which are on/near the perimiter of the point cloud defined by `W`.

Usage

```
VORobj$set_bounds(lb, ub)
```

Arguments

`lb`	a vector of lower bounds
`ub`	a vector of upper bounds

Details

Both `lb` and `ub` can have `length = 1`, in which case the single value will be recycled across dimension. Otherwise, they must have `length = ncol(W)`. Both must be supplied, even if only Internally, fields `lb` and `ub` will be overwritten.

Value

None

set_DADJ *Set the Delaunay graph adjacency*

Description

A Delaunay graph adjacency is required for all ellipsoidal calculations in VorVQ. If the Delaunay adjacency is available from an external source (or calculation), it can be set with this method. Otherwise, one can be computed (and set) via calc_DADJ.

Usage

```
VORobj$set_DADJ
```

Value

None, the field `DADJ` is set internally.

set_params *Set parameters for Voronoi calculations*

Description

Several methods require parameters which can be accessed via this method. The parameters and their defaults (set during a call to `initialize_VOR` are:

`parallel` boolean, whether to perform all calculations in parallel (via RcppParallel). Default = T.

`MVIE_maxiter` The maximum number of iterations allowed when calculating the MVIEs for Voronoi cells. Default = 100.

`MVIE_tol` The relative convergence tolerance for the MVIE routine, which exits when the volume of the computed MVIE changes < this tol. Default = 1e-4.

`MVIE_fix_c0` boolean, whether to allow the MVIE routine to update the MVIE center along with the ellipsoidal rotation. Default = T.

Usage

```
VORobj$set_params(list(param_name = param_value))
```

Arguments

`param_list`	a list with component names in the parameter set defined above, and corresponding desired values.

Value

None

set_vor1_active *Set the active first-order Voronoi cells*

Description

The computations with a VOR object (computing Voronoi centers and ellipsoids) are only performed for "active" cells, which are identified by the prototype index (row index in `W`) which generated the cell. By default the active first-order Voronoi cells are those which contain data mapped to them during a recall of the ANN, as indicated by CADJ (the first-order cells corresponding to the rows of CADJ with with non-zero rowsum). A different set of active cells can be set by calling this method

Usage

```
VORobj$set_vor1_active(vor1_indices)
```

Arguments

`vor1_indices`	a vector containing the indices (rows of `W`, 1-based) of the desired active first-order cells

Value

None, the field `vor1_active` is set internally.

`set_vor1_centers` *Set the centers of first-order Voronoi cells*

Description

The MVIE and Dikin methods require a starting interior point for each active first-order Voronoi polytope. Be default, the prototypes (Voronoi cell generators, as stored in `W`) of each active first-order cell is used. Different interior points can be set with this method. Note: the points passed to this method **must be interior** to each first-order cell; a check will be performed and an error returned if this is not the case. Clear any previously set centers via `clear_vor1_centers`.

Usage

```
VORobj$set_vor1_centers
```

Arguments

`C`	a matrix whose rows specify the active first-order Voronoi cell centers. Must have `nrow(C) = length(vor1_active)`.

Value

None, the field `vor1_centers` is set internally.

`set_vor2_active` *Set the active second-order Voronoi cells*

Description

The computations with a VOR object (computing Voronoi centers and ellipsoids) are only performed for "active" cells, which are identified by the prototype indices (row index in `W`) which generated the cell. By default the active second-order Voronoi cells are those which contain data mapped to them during a recall of the ANN, as indicated by CADJ (those second-order cells corresponding to `CADJ(i,j) > 0`). A different set of active cells can be set by calling this method

Usage

```
VORobj$set_vor2_active(vor2_indices)
```

Arguments

`vor2_indices`	a matrix whose rows contain the `(i,j)` indices (rows of `W`, 1-based) of the desired active second-order cells

Value

None, the field `vor2_active` is set internally.

set_vor2_centers *Set the centers of second-order Voronoi cells*

Description

The MVIE and Dikin methods require a starting interior point for each active second-order Voronoi polytope, which can be set with this method. Note: the points passed to this method **must be interior** to each second-order cell; a check will be performed and an error returned if this is not the case. Clear any previously set centers via `clear_vor2_centers`.

Usage

```
VORobj$set_vor2_centers
```

Arguments

`C`	a matrix whose rows specify the active second-order Voronoi cell centers. Must have `nrow(C) = nrow(vor2_active)`.

Value

None, the field `vor2_centers` is set internally.

SHGR *A SHGRWalk Synthetic Hyperspectral Image Cube*

Description

The SHGRWalk (SHGR, for short) data suite is a collection of Synthetic Hyperspectral 128 x 128 pixel image cubes whose individual pixel "spectra" were sampled from a multi-component Gaussian Mixture Model whose component means were set via a (secondary) Gaussian Random Walk across the synthetic "spectral channels" (the data dimension, `d`). Individual component variances possesses a Toeplitz correlation structure with various noise levels. After sampling the pixels were labeled according to which mixture component (class) they were sampled from, and then organized into the 128 x 128 pixel image such that pixels from the same sampling component occupy contiguous blocks within the overall image. More information about the SHGRWalk data can be found here `need link`.

Usage

```
SHGR
```

Format

For demonstration of Voronoi calculations in higher dimensions a 100-dimensional SHGR cube with 20 distinct classes (each with correlated noise) has been included in the `VorVQ` package. These data are stored in a list variable named `SHGR` with components:

X data matrix whose 16,384 rows represent the 128 x 128 pixels in the image, and 100 columns represent the 100 spectral channels at which the synthetic reflectances were measured

W matrix of 400 learned prototype vectors (in rows, learned from a 20x20 SOM)

CADJ The CADJ matrix resulting from a recall of the data in `X` using the prototypes in `W`. `CADJ` is ordered such that its `(i,j)` element contains the number of data vectors in `X` whose BMU1 = the prototype vector in the `i`-th row and BMU2 = the prototype in the `j`-th row of `W`

`tableau20` *Tableau 20 Color Palette*

Description

Tableau 20 Color Palette

Usage

```
tableau20(col = NULL)
```

Value

a named vector containing Tableau 20 colors

`vis_polytope` *Visualize a (bounded) polytope*

Description

The boundary region of a bounded polytope will be plotted as a set of intersecting line segments.

Usage

```
vis_polytope(
  A,
  b,
  center = NULL,
  add = F,
  fill.col = NULL,
  edge.lty = 1,
  edge.lwd = 1,
  edge.col = "navy",
  vertex.pch = 16,
  vertex.cex = 0.75,
  vertex.col = "navy",
  center.pch = 16,
  center.cex = 0.75,
  center.col = "navy",
  text = NULL,
  text.cex = 0.75,
  text.col = "navy"
)
```

Arguments

A	the constraint matrix (must be 2-dimensional)
b	the constraint RHS
center	a flag denoting the type of center where the polytope label will be plotted as text. Can be one of 1. NULL (default), no center defined 2. a vector (length = 2) giving the (x,y) coords of the center 3. "chebyshev", in which case the Chebyshev center will be calculated and used 4. "analytic", in which case the analytic center will be calculated and used
add	boolean, whether to generate a new plot or add to existing
fill.col	color to fill polygon. Default = NULL means no fill used.
edge.lty	edge linetype
edge.lwd	edge linewidth
edge.col	edge color
vertex.pch	the vertex point style
vertex.cex	the vertex point size, set = 0 to suppress vertex point plotting
vertex.col	the vertex point color
center.pch	the center point style
center.cex	the center point size, set = 0 to suppress center point plotting if center != NULL
center.col	the center point color
text	a string (or numeric value, which will be cast as string) giving the label text which is plotted at (x,y) coords defined by center. Default = NULL (nothing).
text.cex	the text label size
text.col	the text label color

Details

center must be given as something OTHER than NULL if either a point or label is requested to be plotted at the polytope center. If only a text label at the center is desired, set center.cex = 0. If only a point at the center is desired, set label = NULL

Value

nothing, a plot is generated

vis_polytope_constraints *Visualize polytope constraints*

Description

The boundaries x : Ax = b will be plotted as lines

Usage

```
vis_polytope_constraints(
  A,
  b,
  add = F,
  edge.lty = 1,
  edge.lwd = 1,
  edge.col = "magenta"
)
```

Arguments

A	the constraint matrix (must be 2-dimensional)
b	the constraint RHS
add	boolean, whether to generate a new plot or add to existing
edge.lty	edge linetype
edge.lwd	edge linewidth
edge.col	edge color

Details

The plotting parameters *.lty, *.lwd and *.col are passed to R's plot() command.

Value

nothing, a plot will be generated

vis_vor1_annotate *Annotate the first-order Voronoi cells*

Description

Annotate the first-order Voronoi cells

Usage

```
vis_vor1_annotate(
  VOR,
  add = F,
  text = "vorid",
  text.cex = 0.75,
  text.col = "black",
  text.font = 1
)
```

Arguments

VOR	a Voronoi object, instantiated by VOR$new()
add	boolean, whether to generate a new plot or add to existing
text	the text label to be printed in each Voronoi cell. The default is a single string = 'vorid', which prints the ids in $vor1_active at the centers of each active first-order cell. Can also be given as a vector (length = length($vor1_active)) containing a label for each active first-order cell.
text.cex	size of plotted text labels
text.col	color for plotted text labels
text.font	font weight for plotted text labels. Set = 2 for bold.

Details

The text will be plotted at the (x,y) coordinates given by $get_vor1_centers.

Value

nothing, a plot is generated

---

vis_vor1_Dikin *Visualize the first-order Voronoi Dikin Ellipses*

Description

Visualize the first-order Voronoi Dikin Ellipses

Usage

```
vis_vor1_Dikin(
  VOR,
  add = F,
  col = tableau20("green"),
  ell.lwd = 1.5,
  center.pch = 16,
  center.cex = 0.75
)
```

Arguments

VOR	a Voronoi object, instantiated by VOR$new()
add	boolean, whether to generate a new plot or add to existing
col	color for plotting ellipse and its center
ell.lwd	linewidth of ellipse
center.pch	the center point style
center.cex	the center point size, set = 0 to suppress center point plotting

Value

nothing, a plot is generated

---

vis_vor1_MVIE *Visualize the first-order Voronoi MVIEs*

Description

Visualize the first-order Voronoi MVIEs

Usage

```
vis_vor1_MVIE(
  VOR,
  add = F,
  col = tableau20("blue"),
  ell.lwd = 1.5,
  center.pch = 16,
  center.cex = 0.75
)
```

Arguments

VOR	a Voronoi object, instantiated by VOR$new()
add	boolean, whether to generate a new plot or add to existing
col	color for plotting ellipse and its center
ell.lwd	linewidth of ellipse
center.pch	the center point style
center.cex	the center point size, set = 0 to suppress center point plotting

Value

nothing, a plot is generated

vis_vor1_polytopes *Visualize the first-order Voronoi cells generated by prototypes*

Description

Visualize the first-order Voronoi cells generated by prototypes

Usage

```
vis_vor1_polytopes(
  VOR,
  add = F,
  edge.lty = 1,
  edge.lwd = 1,
  edge.col = tableau20("gray"),
  vertex.pch = 16,
  vertex.cex = 0.75,
  vertex.col = tableau20("gray")
)
```

Arguments

VOR	a Voronoi object, instantiated by VOR$new()
add	boolean, whether to generate a new plot or add to existing
edge.lty	edge linetype
edge.lwd	edge linewidth
edge.col	edge color
vertex.pch	the vertex point style
vertex.cex	the vertex point size, set = 0 to suppress vertex point plotting
vertex.col	the vertex point color

Value

nothing, a plot is generated

vis_vor2_annotate *Annotate the second-order Voronoi cells*

Description

Annotate the second-order Voronoi cells

Usage

```
vis_vor2_annotate(
  VOR,
  add = F,
  text = "vorid",
  text.cex = 0.55,
  text.col = "black",
  text.font = 1
)
```

Arguments

VOR	a Voronoi object, instantiated by VOR$new()
add	boolean, whether to generate a new plot or add to existing
text	the text label to be printed in each Voronoi cell. The default is a single string = 'vorid', which prints the ids in $vor2_active at the centers of each active second-order cell. Can also be given as a vector (length = nrow($vor2_active)) containing a label for each active second-order cell.
text.cex	size of plotted text labels
text.col	color for plotted text labels
text.font	font weight for plotted text labels. Set = 2 for bold.

Details

The text will be plotted at the (x,y) coordinates given by $get_vor2_centers.

Value

nothing, a plot is generated

---

vis_vor2_Dikin *Visualize the second-order Voronoi Dikin Ellipses*

Description

Visualize the second-order Voronoi Dikin Ellipses

Usage

```
vis_vor2_Dikin(
  VOR,
  add = F,
  col = tableau20("lightgreen"),
  ell.lwd = 1.5,
  center.pch = 16,
  center.cex = 0.75
)
```

Arguments

VOR	a Voronoi object, instantiated by VOR$new()
add	boolean, whether to generate a new plot or add to existing
col	color for plotting ellipse and its center
ell.lwd	linewidth of ellipse
center.pch	the center point style
center.cex	the center point size, set = 0 to suppress center point plotting

Value

nothing, a plot is generated

---

vis_vor2_MVIE *Visualize the second-order Voronoi MVIEs*

Description

Visualize the second-order Voronoi MVIEs

Usage

```
vis_vor2_MVIE(
  VOR,
  add = F,
  col = tableau20("lightblue"),
  ell.lwd = 1.5,
  center.pch = 16,
  center.cex = 0.75
)
```

Arguments

VOR	a Voronoi object, instantiated by VOR$new()
add	boolean, whether to generate a new plot or add to existing
col	color for plotting ellipse and its center
ell.lwd	linewidth of ellipse
center.pch	the center point style
center.cex	the center point size, set = 0 to suppress center point plotting

Value

nothing, a plot is generated

vis_vor2_polytopes *Visualize the second-order Voronoi cells generated by prototypes*

Description

Visualize the second-order Voronoi cells generated by prototypes

Usage

```
vis_vor2_polytopes(
  VOR,
  add = F,
  edge.lty = 1,
  edge.lwd = 1,
  edge.col = tableau20("lightgray"),
  vertex.pch = 16,
  vertex.cex = 0.75,
  vertex.col = tableau20("lightgray")
)
```

Arguments

VOR	a Voronoi object, instantiated by VOR$new()
add	boolean, whether to generate a new plot or add to existing
edge.lty	edge linetype
edge.lwd	edge linewidth
edge.col	edge color
vertex.pch	the vertex point style
vertex.cex	the vertex point size, set = 0 to suppress vertex point plotting
vertex.col	the vertex point color

Value

nothing, a plot is generated

VOR *The VOR object class*

Description

The VOR object class

Fields

`W` matrix of the prototypes (in rows) which generate the Voronoi tessellation

`nW` the number of prototypes in the tessellation

`d` the dimension of the prototypes = `ncols(W)`

`lb` a vector of dimension-wise lower bounds for the tessellated region, `length = d`

`ub` a vector of dimension-wise upper bounds for the tessellated region, `length = d`

`CADJ` the CADJ adjacency of the learned prototypes

`vor1_active` a vector of indices (rows of W) identifying the first-order Voronoi for which approximating ellipsoids will be computed. Defaults to the active first-order cells in the tessellation (those which contain at least one data observation after the ANN recall, as indicated by CADJ).

`vor2_active` a matrix whose rows contain the (i,j) indices identifying the second-order Voronoi cells for which approximating ellipsoids will be computed. Defaults to the active second-order cells in the tessellation (those which contain at least one data observation after the ANN recall, as indicated by CADJ).

`GADJ` the Gabriel graph adjacency matrix

`DADJ` the Delaunay graph adjacency matrix

`vor1_centers` matrix whose rows store the Chebyshev centers of the first-order Voronoi cells listed in `vor1_active`

`vor2_centers` matrix whose rows store the Chebyshev centers of the second-order Voronoi cells listed in `vor2_active`

`vor1_MVIE_c` matrix whose rows store the MVIE centers of the first-order Voronoi cells listed in `vor1_active`

`vor1_MVIE_E` cube whose slices store the MVIE ellipsoidal rotation matrices of the first-order Voronoi cells listed in `vor1_active`

`vor1_MVIE_logdet` vector whose elements store the log-determinant of MVIE ellipsoidal rotation matrices of the first-order Voronoi cells listed in `vor1_active`

`vor1_MVIE_volratio` vector whose elements store the proportional volume of the MVIEs of the first-order Voronoi cells listed in `vor1_active`

`vor1_MVIE_status` return flag from running the MVIE routine for each of the first-order Voronoi cells listed in `vor1_active`

`vor2_MVIE_c` matrix whose rows store the MVIE centers of the second-order Voronoi cells listed in `vor2_active`

`vor2_MVIE_E` cube whose slices store the MVIE ellipsoidal rotation matrices of the second-order Voronoi cells listed in `vor2_active`

`vor2_MVIE_logdet` vector whose elements store the log-determinant of MVIE ellipsoidal rotation matrices of the second-order Voronoi cells listed in `vor2_active`

`vor2_MVIE_volratio` vector whose elements store the proportional volume of the MVIEs of the second-order Voronoi cells listed in `vor2_active`

`vor2_MVIE_status` return flag from running the MVIE routine for each of the second-order Voronoi cells listed in `vor2_active`

`vor1_Dikin_E` cube whose slices store the Dikin ellipsoidal rotation matrices of the first-order Voronoi cells listed in `vor1_active`

`vor2_Dikin_E` cube whose slices store the Dikin ellipsoidal rotation matrices of the second-order Voronoi cells listed in `vor2_active`

`vis_par` List containing the `par()` parameters set during a call to the the `vis_vor_*` functions.

Methods

Each class method has its own documentation, accessible via `?VorVQ::<method_name>`. For completeness, the list is repeated here in entirety. Additional functionality for visualizing a two-dimensional VOR object is available through the `vis_*` functions. See their documentation for more information.

`VOR$new` Instantiate a VOR object

`initialize_VOR` Initialize the VOR object with the prototypes and CADJ matrix from ANN learning

`set_bounds` Set the upper and lower bounds of the tessellated region

`set_vor1_active` Set the indices identifying the active first-order Voronoi cells

`set_vor2_active` Set the indices identifying the active second-order Voronoi cells

`get_params` Get the parameters used for various methods of the VOR object

`set_params` Set the parameters used for various methods of the VOR object

`calc_GADJ` Calculate the Gabriel adjacency matrix of the tessellation

`calc_DADJ` Calculate the Delaunay adjacency matrix of the tessellation

`set_DADJ` Set the Delaunay adjacency matrix of the tessellation, if obtained elsewhere

`get_params` Get the parameters used for various methods of the VOR object

`get_vor1_polytope` Get the polytope definition of a first-order Voronoi cell

`get_vor2_polytope` Get the polytope definition of a second-order Voronoi cell

`calc_vor1_centers` Compute the Chebhyshev centers of the active first-order Voronoi cells

`calc_vor2_centers` Compute the Chebhyshev centers of the active second-order Voronoi cells

`set_vor1_centers` Set the centers of the active first-order Voronoi cells

`set_vor2_centers` Set the centers of the active second-order Voronoi cells

`get_vor1_centers` Get the centers of the active first-order Voronoi cells

`get_vor2_centers` Get the centers of the active second-order Voronoi cells

`clear_vor1_centers` Clear the centers of the active first-order Voronoi cells

`clear_vor2_centers` Clear the centers of the active second-order Voronoi cells

`calc_vor1_MVIE` Compute the Maximum Volume Inscribed Ellipsoids (MVIE) of the active first-order Voronoi cells

`calc_vor2_MVIE` Compute the Maximum Volume Inscribed Ellipsoids (MVIE) of the active second-order Voronoi cells

`calc_vor1_Dikin` Compute the Dikin Ellipsoids of the active first-order Voronoi cells

`calc_vor2_Dikin` Compute the Dikin Ellipsoids of the active second-order Voronoi cells

`calc_all` Compute all Voronoi quantities

`save` Save a VOR object to disk

`load` Load a previously saved VOR object from disk

`as_list` Convert and return all fields of a VOR object to an R list.

`load_list` Populate an instance of a VOR object from an R list.

vor1_polytope — *First Order Voronoi Cell Polytope Returns the polytope definition Ax <= b of a single Voronoi cell*

Description

First Order Voronoi Cell Polytope Returns the polytope definition Ax <= b of a single Voronoi cell

Usage

```
vor1_polytope(
  W,
  iidx,
  DADJ = NULL,
  lb = NULL,
  ub = NULL,
  rmv_redundancies = FALSE
)
```

Arguments

W	matrix of prototypes in rows
iidx	Index of Voronoi cell, 1-based.
DADJ	(optional) - Delaunay adjacency. If used, the H-representation of the returned polytope will be tight, only including constraints from Delaunay neighbors.
lb	(optional) - Global lower bound of data. If given, the H-representation of the returned polytope will include this as a lower bound.
ub	(optional) - same as xlb, but for upper bounds. Both must be given in order for the global bounds to be appended to the polytope representation.
rmv_redundancies	boolean, whether to check the system for redundancies, and remove any found

Value

a list with components A, b, cid

vor2_polytope — *Second Order Voronoi Cell Polytope Returns the polytope definition Ax <= b of a second order Voronoi cell*

Description

Second Order Voronoi Cell Polytope Returns the polytope definition Ax <= b of a second order Voronoi cell

Usage

```
vor2_polytope(
  W,
  iidx,
  jidx,
  DADJ = NULL,
  lb = NULL,
  ub = NULL,
  rmv_redundancies = FALSE
)
```

Arguments

W	matrix of prototypes in rows
iidx	Index of 1st Voronoi cell
jidx	Index of 2nd Voronoi cell
DADJ	Delaunay adjacency. The H-representation of the returned polytope will be tight, only including constraints from Delaunay neighbors.
lb	- Global lower bound of data. If given, the H-representation of the returned polytope will include this as a lower bound.
ub	- same as xlb, but for upper bounds. Both must be given in order for the global bounds to be appended to the polytope representation.
rmv_redundancies	boolean, whether to check the system for redundancies, and remove any found

Value

a list with components A, b, cid

Index