sBMH

July 21, 2015

sBMH	*Function to identify the best-matching hexagons/rectangles for the input data*

Description

sBMH is supposed to identify the best-matching hexagons/rectangles (BMH) for the input data.

Usage

```
sBMH(sMap, data, which_bmh = c("best", "worst", "all"))
```

Arguments

sMap	an object of class "sMap" or a codebook matrix
data	a data frame or matrix of input data
which_bmh	which BMH is requested. It can be a vector consisting of any integer values from [1, nHex]. Alternatively, it can also be one of "best", "worst" and "all" choices. Here, "best" is equivalent to 1, "worst" for $nHex$, and "all" for $seq(1, nHex)$

Value

a list with following components:

- bmh: the requested BMH matrix of dlen x length(which_bmh), where dlen is the total number of rows of the input data
- qerr: the corresponding matrix of quantization errors (i.e., the distance between the input data and their BMH), with the same dimensions as "bmh" above
- mqe: the mean quantization error for the "best" BMH
- call: the call that produced this result

Note

"which_bmh" upon request can be a vector consisting of any integer values from [1, nHex]

See Also

sPipeline

Examples

```
# 1) generate an iid normal random matrix of 100x10
data <- matrix( rnorm(100*10,mean=0,sd=1), nrow=100, ncol=10)

# 2) from this input matrix, determine nHex=5*sqrt(nrow(data))=50,
# but it returns nHex=61, via "sHexGrid(nHex=50)", to make sure a supra-hexagonal grid
sTopol <- sTopology(data=data, lattice="hexa", shape="suprahex")

# 3) initialise the codebook matrix using "uniform" method
sI <- sInitial(data=data, sTopol=sTopol, init="uniform")

# 4) define trainology at "rough" stage
sT_rough <- sTrainology(sMap=sI, data=data, stage="rough")

# 5) training at "rough" stage
sM_rough <- sTrainBatch(sMap=sI, data=data, sTrain=sT_rough)

# 6) define trainology at "finetune" stage
sT_finetune <- sTrainology(sMap=sI, data=data, stage="finetune")

# 7) training at "finetune" stage
sM_finetune <- sTrainBatch(sMap=sM_rough, data=data, sTrain=sT_rough)

# 8) find the best-matching hexagons/rectangles for the input data
response <- sBMH(sMap=sM_finetune, data=data, which_bmh="best")
```