sTrainology

June 24, 2016

---

sTrainology   *Function to define trainology (training environment)*

Description

sTrainology is supposed to define the train-ology (i.e., the training environment/parameters). The trainology here refers to the training algorithm, the training stage, the stage-specific parameters (alpha type, initial alpha, initial radius, final radius and train length), and the training neighbor kernel used. It returns an object of class "sTrain".

Usage

```
sTrainology(sMap, data, algorithm = c("batch", "sequential"),
stage = c("rough", "finetune", "complete"), alphaType = c("invert",
"linear", "power"), neighKernel = c("gaussian", "bubble",
"cutgaussian",
"ep", "gamma"))
```

Arguments

sMap	an object of class "sMap" or "sInit"
data	a data frame or matrix of input data
algorithm	the training algorithm. It can be one of "sequential" and "batch" algorithm
stage	the training stage. The training can be achieved using two stages (i.e., "rough" and "finetune") or one stage only (i.e., "complete")
alphaType	the alpha type. It can be one of "invert", "linear" and "power" alpha types
neighKernel	the training neighbor kernel. It can be one of "gaussian", "bubble", "cutgaussian", "ep" and "gamma" kernels

Value

an object of class "sTrain", a list with following components:

- algorithm: the training algorithm
- stage: the training stage
- alphaType: the alpha type

1

- `alphaInitial`: the initial alpha
- `radiusInitial`: the initial radius
- `radiusFinal`: the final radius
- `neighKernel`: the neighbor kernel
- `call`: the call that produced this result

Note

Training stage-specific parameters:

- "radiusInitial": it depends on the grid shape and training stage
  - For "sheet" shape: it equals $max(1, ceiling(max(xdim, ydim)/8))$ at "rough" or "complete" stage, and $max(1, ceiling(max(xdim, ydim)/32))$ at "finetune" stage
  - For "suprahex" shape: it equals $max(1, ceiling(r/2))$ at "rough" or "complete" stage, and $max(1, ceiling(r/8))$ at "finetune" stage
- "radiusFinal": it depends on the training stage
  - At "rough" stage, it equals $radiusInitial/4$
  - At "finetune" or "complete" stage, it equals 1
- "trainLength": how many times the whole input data are set for training. It depends on the training stage and training algorithm
  - At "rough" stage, it equals $max(1, 10 * trainDepth)$
  - At "finetune" stage, it equals $max(1, 40 * trainDepth)$
  - At "complete" stage, it equals $max(1, 50 * trainDepth)$
  - When using "batch" algorithm and the trainLength equals 1 according to the above equation, the trainLength is forced to be 2 unless $radiusInitial$ equals $radiusFinal$
  - Where $trainDepth$ is the training depth, defined as $nHex/dlen$, i.e., how many hexagons/rectanges are used per the input data length (here $dlen$ refers to the number of rows)

See Also

[sInitial](#)

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 different stages
# 4a) define trainology at "rough" stage
sT_rough <- sTrainology(sMap=sI, data=data, stage="rough")
# 4b) define trainology at "finetune" stage
sT_finetune <- sTrainology(sMap=sI, data=data, stage="finetune")
# 4c) define trainology using "complete" stage
sT_complete <- sTrainology(sMap=sI, data=data, stage="complete")
```