1. selforgmap

Self-organizing map

Self-organizing maps learn to cluster data based on similarity, topology, with a preference (but no guarantee) of assigning the same number of instances to each class. Self-organizing maps are used both to cluster data and to reduce the dimensionality of data.

Self-organizing feature maps (SOFM) learn to classify input vectors according to how they are grouped in the input space. They differ from competitive layers in that neighboring neurons in the self-organizing map learn to recognize neighboring sections of the input space. Thus, self-organizing maps learn both the distribution (as do competitive layers) and topology of the input vectors they are trained on.

The neurons in the layer of an SOFM are arranged originally in physical positions according to a topology function. The function [gridtop](https://localhost:31515/static/help/deeplearning/ref/gridtop.html), [hextop](https://localhost:31515/static/help/deeplearning/ref/hextop.html), or [randtop](https://localhost:31515/static/help/deeplearning/ref/randtop.html) can arrange the neurons in a grid, hexagonal, or random topology. Distances between neurons are calculated from their positions with a distance function. There are four distance functions, [dist](https://localhost:31515/static/help/deeplearning/ref/dist.html), [boxdist](https://localhost:31515/static/help/deeplearning/ref/boxdist.html), [linkdist](https://localhost:31515/static/help/deeplearning/ref/linkdist.html), and [mandist](https://localhost:31515/static/help/deeplearning/ref/mandist.html). Link distance is the most common. These topology and distance functions are described in [Topologies (gridtop, hextop, randtop)](https://localhost:31515/static/help/deeplearning/ug/cluster-with-self-organizing-map-neural-network.html#bss4b_l-2) and [Distance Functions (dist, linkdist, mandist, boxdist)](https://localhost:31515/static/help/deeplearning/ug/cluster-with-self-organizing-map-neural-network.html#bss4b_l-3).

selforgmap(dimensions,coverSteps,initNeighbor,topologyFcn,distanceFcn) takes these arguments,

|  |  |
| --- | --- |
| Dimensions | Row vector of dimension sizes (default = [8 8]) |
| coverSteps | Number of training steps for initial covering of the input space (default = 100) |
| initNeighbor | Initial neighborhood size (default = 3) |
| topologyFcn | Layer topology function (default = 'hextop') |
| distanceFcn | Neuron distance function (default = 'linkdist') |

and returns a self-organizing map.

1. configure

Configure network inputs and outputs to best match input and target data

Configuration is the process of setting network input and output sizes and ranges, input preprocessing settings and output postprocessing settings, and weight initialization settings to match input and target data.

Configuration must happen before a network’s weights and biases can be initialized. Unconfigured networks are automatically configured and initialized the first time[train](https://localhost:31515/static/help/deeplearning/ref/train.html) is called. Alternately, a network can be configured manually either by calling this function or by setting a network’s input and output sizes, ranges, processing settings, and initialization settings properties manually.

net = configure(net,x,t) takes input data x and target data t, and configures the network’s inputs and outputs to match.

net = configure(net,x) configures only inputs.

1. train

Train shallow neural network

[collapse all in page](javascript:void(0);)

For deep learning with convolutional or LSTM neural networks, see [trainNetwork](https://localhost:31515/static/help/deeplearning/ref/trainnetwork.html) instead.

[trainedNet,tr] = train(net,X,T,Xi,Ai,EW,Name,Value)

### net — Input network network object

Input network, specified as a network object. To create a network object, use for example, [feedforwardnet](https://localhost:31515/static/help/deeplearning/ref/feedforwardnet.html) or [narxnet](https://localhost:31515/static/help/deeplearning/ref/narxnet.html).

### X — Network inputs matrix | cell array | composite data | gpuArray

Network inputs, specified as an R-by-Q matrix or an Ni-by-TS cell array, where

* R is the input size
* Q is the batch size
* Ni = net.numInputs
* TS is the number of time steps

train arguments can have two formats: matrices, for static problems and networks with single inputs and outputs, and cell arrays for multiple timesteps and networks with multiple inputs and outputs.

### 'CheckpointFile' — Checkpoint file '' (default) | character vector

Checkpoint file, specified as a character vector.

The value for 'CheckpointFile' can be set to a filename to save in the current working folder, to a file path in another folder, or to an empty string to disable checkpoint saves (the default value).

### 'CheckpointDelay' — Checkpoint delay 60 (default) | nonnegative integer

Checkpoint delay, specified as a nonnegative integer.

The optional parameter 'CheckpointDelay' limits how often saves happen. Limiting the frequency of checkpoints can improve efficiency by keeping the amount of time saving checkpoints low compared to the time spent in calculations. It has a default value of 60, which means that checkpoint saves do not happen more than once per minute. Set the value of 'CheckpointDelay' to 0 if you want checkpoint saves to occur only once every epoch.