maputils: Popsom Python Interface Module

This module is modeled after the R interface to popsom. For a more object-oriented approach see the documentation on the sklearn API.

map_build

Description

Construct a self-organizing map and return an object of class `map`.

Usage

```
map_build(data, labels = None, xdim = 10, ydim = 5,
          alpha = 0.3, train = 1000, normalize = False,
          seed = None, minimal = False)
```

Arguments

- **data**: A dataframe where each row contains an unlabeled training instance.
- **labels**: A vector or dataframe with one label for each observation in data.
- **xdim**: The x-dimension of the map.
- **ydim**: The y-dimension of the map.
- **alpha**: The learning rate, should be a value greater than zero and less or equal to one.
- **train**: The number of training iterations.
- **normalize**: Boolean switch indicating whether or not to normalize the data.
- **seed**: A seed value for repeatability of random initialization and selection.
- **minimal**: Boolean switch indicating whether to build a `map_minimal` or `map` object.

Details

The function `map_build` constructs an object of type `map`. The object contains two models: 1. **Self-organizing map model:** Expressed through its trained neurons. The quality of the fit can be ascertained by the `convergence` measure. 2. **Cluster model:** Expressed by the discovered centroids. The quality of this model is determined by the `map convergence`, `within cluster sum of squares` (wcss), and the `between cluster sum of squares` (bcss).

Value

An object of type `map` with the following member fields:

- **data**: Data frame containing the possibly normalized training data.
- **labels**: Vector of labels, one for each observation in data or `NULL` if no labels were given.

- **xdim**: The x-dimension of the neuron map.
- **ydim**: The y-dimension of the neuron map.
- **alpha**: The given learning rate for the neural network.
- **train**: The number of training iterations applied to the neural network.
- **neurons**: A list (data frame) of neurons for the network. Its dimensionality is the same as the training data. Two useful formulas:
  - To compute the (x,y) coordinate for the neuron in row `rowix`:
    ```
    x = (rowix - 1) % map['xdim'] + 1
    y = (rowix - 1) // map['xdim'] + 1
    ```
  - To compute the row index from the (x,y)-position:
    ```
    rowix = x + (y - 1) * map['xdim']
    ```
- **heat**: The representation of the map used for the `starburst` plot.
- **fitted.obs**: A list of indexes of the best matching neuron for each observation (row index into the `neurons` data frame).
- **centroids**: A data frame of (x,y)-locations indicating where each centroid is located. (Each centroid points to itself.)
- **unique.centroids**: A vector of unique centroid (x,y)-locations. (Hint: the length of this vector equals the number of clusters.)
- **centroid.labels**: A data frame mapping (x,y)-locations of actual centroids to their labels. If the training data is unlabeled, popsom generates a label for each centroid.
- **label.to.centroid**: A lookup table (hash) mapping labels to centroid indexes. Note that a label may be associated with multiple centroids.
- **centroid.obs**: A vector of lists of observations per centroid (indexed by the centroid number from `unique_centroids`). Each list contains row numbers from the `data` dataframe.
- **convergence**: A quality measure indicating how well the map fits the training data.
- **wcss**: The average "within cluster sum of squares" (variance within clusters).
- **bcss**: The "between cluster sum of squares" (variance between cluster centroids).

Notes

- If the `minimal` switch is set to `True`, then a `map_minimal` object is returned. This object only contains the trained neurons and the training parameters. Note that none of the more involved functions will work with this type of object.
- If your training data is unlabeled, popsom will automatically generate a label for each discovered centroid.

References

- **VSOM: Efficient, Stochastic Self-Organizing Map Training**
  Lutz Hamel, *Intelligent Systems Conference (IntelliSys) 2018*, K. Arai et

al. (Eds.): Intelligent Systems and Applications, Advances in Intelligent Systems and Computing 869, pp. 805-821, Springer, 2018.

- **Self-Organizing Map Convergence**
  Robert Tatoian and Lutz Hamel, *Proceedings of the 2016 International Conference on Data Mining (DMIN'16)*, pp. 92-98, July 25-28, 2016, Las Vegas, Nevada, USA, ISBN: 1-60132-431-6, CSREA Press.
- **Evaluating Self-Organizing Map Quality Measures as Convergence Criteria**
  Gregory Breard and Lutz Hamel, *Proceedings of the 2018 International Conference on Data Science (ICDATA'18)*, Robert Stahlbock, Gary M. Weiss, Mahmoud Abou-Nasr (Eds.), ISBN: 1-60132-481-2, pp. 86-92, CSREA Press, 2018.
- **SOM Quality Measures: An Efficient Statistical Approach**
  Lutz Hamel, *Proceedings of the 11th International Workshop WSOM 2016*, Houston, Texas, USA, E. Merenyi et al. (Eds.), Advances in Self-Organizing Maps and Learning Vector Quantization, Advances in Intelligent Systems and Computing 428, Springer, pp. 49-59, DOI 10.1007/978-3-319-28518-4_4, 2016.

Examples

```
import pandas as pd
from popsom7 import maputils
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

## build a map
m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 10000, seed = 42)

## look at the characteristics of the map
maputils.map_summary(m)

## plot the map
maputils.map_starburst(m)
```

---

map_convergence

Description

Evaluate the quality of a SOM using embedding accuracy and estimated topographical accuracy.

Usage

```
map_convergence(map, conf_int = 0.95, k = 50, verb = True, ks = True)
```

Arguments

- **map**: An object of type `map`.
- **conf.int**: The confidence interval of the quality assessment.
- **k**: Number of samples to use in the computation of the estimated topographical accuracy.
- **verb**: If `True`, reports the two convergence components separately; otherwise, a linear combination of the two indices is reported.
- **ks**: If `True`, uses the Kolmogorov-Smirnov convergence test; otherwise, a test based on variance and means is performed.

Value

A single value or a pair of values: 1. Embedding accuracy 2. Estimated topographic accuracy

The structure of the return value depends on the `verb` switch.

References

- **SOM Quality Measures: An Efficient Statistical Approach**
  Lutz Hamel, *Proceedings of the 11th International Workshop WSOM 2016*, Houston, Texas, USA, E. Merenyi et al. (Eds.), Advances in Self-Organizing Maps and Learning Vector Quantization, Advances in Intelligent Systems and Computing 428, Springer, pp. 49-59, DOI 10.1007/978-3-319-28518-4_4, 2016.

Examples

```
import pandas as pd
from popsom7 import maputils
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

## build a map
m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 1000)

## map quality
maputils.map_convergence(m)
```

---

map_fitted

Description

Computes a vector of labels assigned to each of the observations in the training data through the constructed cluster model. If the training data is unlabeled, machine-generated labels are used.

Usage

```
map_fitted(map)
```

Arguments

- **map**: An object of type `map`.

Value

A vector of predicted labels, one for each observation in the training data.

Examples

```
import pandas as pd
from popsom7 import maputils
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 10000)

fitted_labels = maputils.map_fitted(m)
```

---

map_marginal

Description

Generate a plot that shows the marginal probability distribution of the neurons and data.

Usage

```
map_marginal(map, marginal)
```

Arguments

- **map**: An object of type `map`.
- **marginal**: The name of a training data dimension or index.

Examples

```
import pandas as pd
from popsom7 import maputils
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

## build a map
m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 10000)

## display marginal distribution of dimension 1
maputils.map_marginal(m, 1)
```

---

map_position

Description

Compute the (x,y)-positions of points on the map.

Usage

```
map_position(map, points)
```

Arguments

- **map**: An object of type `map`.
- **points**: A data frame of points to be mapped.

Value

A data frame with (x,y)-positions. The data frame has two columns: - **x-dim**: The x-position of the corresponding point in the `points` data frame. - **y-dim**: The y-position of the corresponding point in the `points` data frame.

Examples

```
import pandas as pd
from popsom7 import maputils
from sklearn import datasets
```

```
iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 10000)

positions = maputils.map_position(m, X)
```

---

map_predict

Description

Compute classification labels for points in a given data frame using the underlying clustering model. If the training data is unlabeled, machine-generated labels are used.

Usage

```
map_redict(map, points)
```

Arguments

- **map**: An object of type `map`.
- **points**: A data frame of points to be classified.

Value

A data frame with classification results. The data frame has two columns: - **Label**: The assigned label to the observation at the same row in the `points` data frame. - **Confidence**: A confidence value assigned to the label prediction.

Examples

```
import pandas as pd
from popsom7 import maputils
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 10000)

y_predict = maputils.map_predict(m, X)
```

---

map_significance

Description

Computes the relative significance of each feature and plots it.

Usage

```
map_significance(map, graphics = False, feature_labels = True)
```

Arguments

- **map**: An object of type `map`.
- **graphics**: A switch that controls whether a plot is generated or not.
- **feature.labels**: A switch to allow the plotting of feature names vs. feature indices.

Value

If `graphics = False`, a vector containing the significance for each feature is returned.

Note: A Bayesian approach is used to compute the relative significance of features based on variance.

References

- **Bayesian Probability Approach to Feature Significance for Infrared Spectra of Bacteria**
  Lutz Hamel, Chris W. Brown, *Applied Spectroscopy, Volume 66, Number 1, 2012.*

Examples

```
import pandas as pd
from popsom7 import maputils
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 10000)

## Display the relative feature significance graphically
maputils.map_significance(m)
```

map_starburst

Usage

```
map_starburst(map)
```

Arguments

- **map**: An object of type `map`.

Description

Generate a starburst representation of the clusters on the heat map for the self-organizing map model.

References

- **Improved Interpretability of the Unified Distance Matrix with Connected Components**
  Lutz Hamel and Chris W. Brown, *Proceedings of the 7th International Conference on Data Mining (DMIN'11)*, July 18-21, 2011, Las Vegas, Nevada, USA, ISBN: 1-60132-168-6, pp. 338-343, CSREA Press, 2011.

Examples

```
import pandas as pd
from popsom7 import maputils
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 10000)

maputils.map_starburst(m)
```

---

map_summary

Description

Generate a summary object for `map` objects.

Usage

```
map_summary(map, verb = True)
```

Arguments

- **map**: An object of type `map`.
- **verb**: A switch controlling the output.

Value

An object of type `summary_map` containing two structures: - **training.parameters**: A dataframe containing the parameters used to train the map. - **quality.assessments**: A dataframe containing the quality assessments of the map. In particular, it includes: - **convergence**: A linear combination of variance capture and topographic fidelity. A value close to 1 indicates a converged map. - **separation**: Computed as `1 - wcss / bcss`, where a value close to 1 indicates well-separated clusters.

If `verb` is `True`, the `summary_map` object is formatted and printed to the screen; otherwise, it is returned as a data structure.

References

- **Self-Organizing Map Convergence**
  Robert Tatoian and Lutz Hamel, *Proceedings of the 2016 International Conference on Data Mining (DMIN'16)*, pp. 92-98, July 25-28, 2016, Las Vegas, Nevada, USA, ISBN: 1-60132-431-6, CSREA Press.

Examples

```python
import pandas as pd
from popsom7 import maputils
from sklearn import datasets

iris = datasets.load_iris()
X = pd.DataFrame(iris.data, columns=iris.feature_names)
y = pd.DataFrame(iris.target_names[iris.target],columns=['species'])

## build a map
m = maputils.map_build(X, y, xdim = 15, ydim = 10, train = 10000)

## compute a summary object and display it
s = maputils.map_summary(m, verb = False)
print(s)
```