TDA for the user

Seminar 1

10/01/23

Eduardo Paluzo-Hidalgo

Steps:

1. Data manipulation

2. Persistent homology
    2.1 Filtration
    2.2 Vectorization

3. Machine Learning

ANACONDA

(https://www.anaconda.com/)

First: environment

Second:

Terminal

Notebook

Coding

Library installation

https://github.com/Cimagroup/vectorization-maps

In the downloaded folder:

```
pip install -r requirements.txt

pip install .
```

Steps:

1. Data manipulation

Libraries:
numpy (https://numpy.org/doc/stable/)
pandas (https://pandas.pydata.org/)

Steps:

1. Data manipulation

Libraries:
numpy (https://numpy.org/doc/stable/)
pandas (https://pandas.pydata.org/)

Dataset: (X, Y)

```
from sklearn import datasets
```

```
iris = datasets.load_iris()
X = iris.data
y = iris.target
```

2. Persistent homology

2.1 Filtration

ripser

```
from ripser import ripser
```

```
vietorisRips=ripser(X)
diagrams = vietorisRips["dgms"]
```

Gudhi

```
import gudhi as gd
```

```
rips_complex = gd.RipsComplex(points=X)
simplex_tree = rips_complex.create_simplex_tree(max_dimension=1)
diag = simplex_tree.persistence()
```

Giotto-tda (pip install giotto-tda)

```
from gtda.homology import VietorisRipsPersistence
```

```
VR = VietorisRipsPersistence(homology_dimensions=[0, 1])
diagrams = VR.fit_transform([X])
```

2. Persistent homology

2.2 Vectorization

Vectorization Method
Persistence Statistics
Entropy Summary
Algebraic Functions
Tropical Coordinates
Complex Polynomial
Betti Curve
Lifespan Curve
Persistence Landscape
Persistence Silhouette
Persistence Image
Template Function
Adaptive Template System
ATOL

```
import vectorization as vect
```

```
vect.GetPersStats(diagrams[1])
```

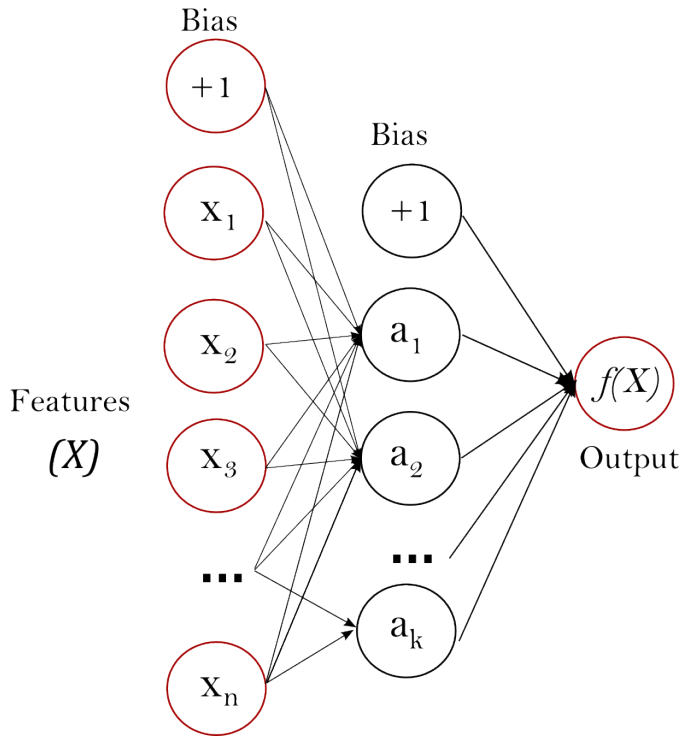
```
array([4.12347669e-01, 4.53902274e-01, 1.19008202e-01, 1.26127747e-01,
       3.87298346e-01, 4.47213590e-01, 1.21416613e-01, 1.22937322e-01,
       6.15283564e-01, 6.32737875e-01, 3.00000012e-01, 3.31662476e-01,
       3.31318960e-01, 3.66854250e-01, 4.52735573e-01, 4.89791572e-01,
       5.47722578e-01, 5.91607988e-01, 4.33124971e-01, 1.21761197e-01,
       4.17255968e-01, 1.10457599e-01, 6.24010719e-01, 3.16227764e-01,
       3.46410155e-01, 4.69041586e-01, 5.55871075e-01, 4.15546058e-02,
       2.89683374e-02, 3.16624641e-02, 4.84748036e-02, 9.55199599e-02,
       1.16258264e-02, 1.54347122e-02, 6.39095157e-02, 8.66250396e-02,
       3.10000000e+01, 3.19163469e+00])
```

```
vect.GetAlgebraicFunctions(diagrams[1])
```

```
array([5.45227712e-01, 5.76774053e-01, 9.74981119e-05, 9.41296958e-05,
       1.04886174e-01])
```

3. Machine Learning

1. Split the dataset in training, (validation), and test
2. Fit machine learning model using the vectorization method as input
3. Evaluate on test set



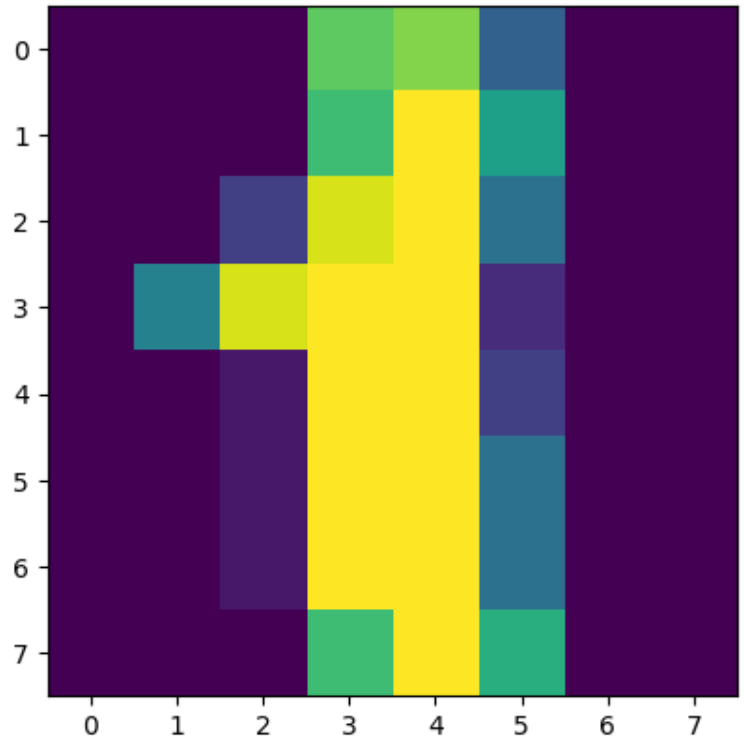
Example

```
from sklearn.datasets import load_digits
from ripser import ripser, lower_star_img
import numpy as np
```

```
digits = load_digits()
```

```
X=digits["data"]
y=digits["target"]
```

```
n_image = 5
plt.imshow(np.reshape(X[1],(8,8)))
plt.show()
```



Example

```
dgms = list()
for i in range(len(X)):
    dgm = lower_star_img(np.reshape(X[i],(8,8)))[:-1]
    dgms.append(dgm)
```

```
import vectorization as vect
```

```
perstats = list()
for i in range(len(X)):
    ps = vect.GetPersStats(dgms[i])
    perstats.append(ps)
```

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score
```

```
X_train, X_test, y_train, y_test = train_test_split(
...     perstats, y, test_size=0.33, random_state=42)
```

```
clf = DecisionTreeClassifier(max_depth=None, min_samples_split=3)
scores = cross_val_score(clf, X_train, y_train, cv=5)
scores.mean()
```

```
0.31419087136929463
```

```
from sklearn.neural_network import MLPClassifier
```

```
clf = MLPClassifier(solver = "adam",hidden_layer_sizes=(64, 32), random_state=1,max_iter=1000)
scores = cross_val_score(clf, X_train, y_train, cv=5)
scores.mean()
```

```
0.3549273858921162
```

Example

```
dgms = list()
for i in range(len(X)):
    dgm = lower_star_img(np.reshape(X[i],(8,8)))[:-1]
    dgms.append(dgm)
```

```
import vectorization as vect
```

```
perstats = li
for i in rang
    ps = vect
    perstats.
```

```
from sklearn.
from sklearn.
from sklearn.
```

```
X_train, X_te
...     perst
```

```
clf = Decisio
scores = cros
scores.mean()
```

```
0.31419087136929463
```

```
from sklearn.neural_network import MLPClassifier
```

```
clf = MLPClassifier(solver = "adam",hidden_layer_sizes=(64, 32), random_state=1,max_iter=1000)
scores = cross_val_score(clf, X_train, y_train, cv=5)
scores.mean()
```

```
0.3549273858921162
```

Gracias