# PyMinimax

*Release 0.1.2*

**beginnerSC**

**Aug 30, 2021**

# TABLE OF CONTENTS

**Package Status**

PyMinimax is a Python implementation of Bien and Tibshirani's paper "Hierarchical Clustering With Prototypes via Minimax Linkage"[1]. This is an agglomerative hierarchical clustering algorithm available in the protoclust R package[2] but not currently in SciPy nor scikit-learn. It has a great advantage over classical hierarchical clustering methods that in each cluster one prototype is selected from the original data. Thus the results are better interpretable.

PyMinimax has a SciPy-compatible API. Users who are familiar with the scipy.cluster.hierarchy module will find it easy to use.

[1] J. Bien and R. Tibshirani. Hierarchical clustering with prototypes via minimax linkage. Journal of the American Statistical Association, 106(495), 2011, 1075-1084.

[2] J. Bien and R. Tibshirani. Package 'protoclust', 2015.

CHAPTER

ONE

# QUICK START

## 1.1 Installation

The recommended way to install PyMinimax is using pip:

```
pip install pyminimax
```

Or if you have installed PyMinimax before, please update to the latest version:

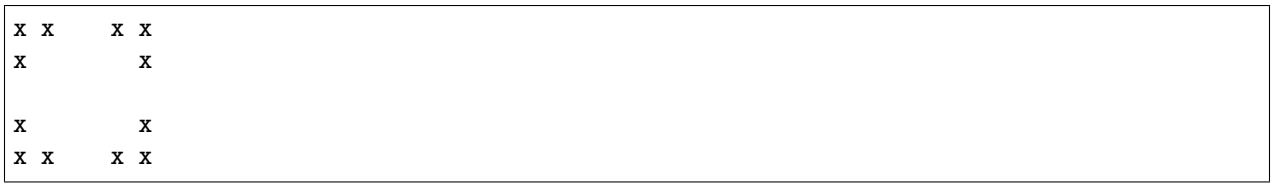
```
pip install --upgrade pyminimax
```

PyMinimax runs on any platform with Python 3 and SciPy installed.

## 1.2 Usage

The most important function in PyMinimax is `pyminimax.minimax`, and by default its usage is the same as the hierarchical clustering methods in SciPy, say `scipy.cluster.hierarchy.complete`. Here we demonstrate with an example from the SciPy documentation. First consider a dataset of $n = 12$ points:

[1]:
```
X = [[0, 0], [0, 1], [1, 0],
     [0, 4], [0, 3], [1, 4],
     [4, 0], [3, 0], [4, 1],
     [4, 4], [3, 4], [4, 3]]
```



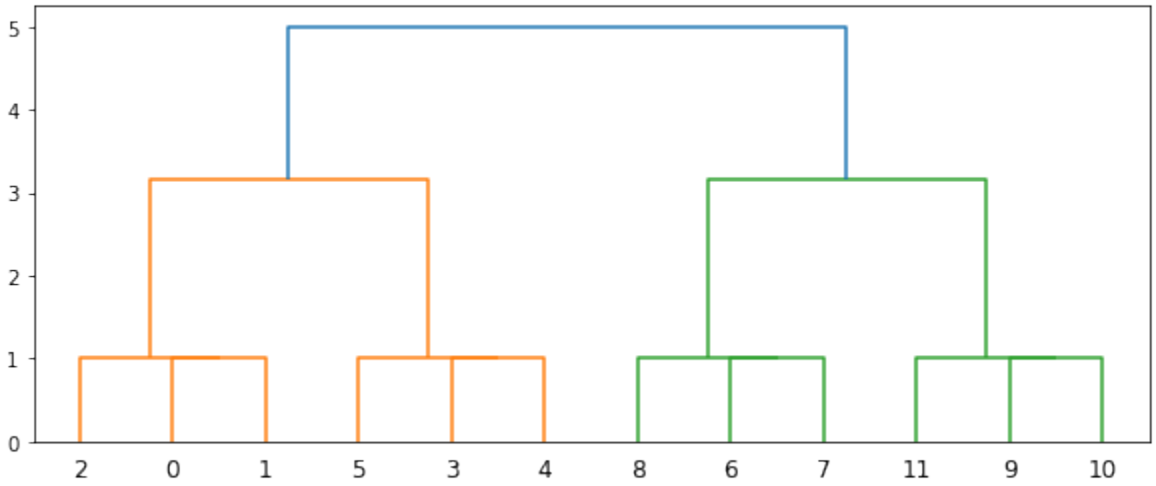
The minimax function takes a flattened distance matrix of the data as an argument, which can be computed by `scipy.spatial.distance.pdist`. By default, the return value of minimax has the same format as that of `scipy.cluster.hierarchy.linkage`. This is an $(n-1)$ by 4 matrix keeping the clustering result, called the *linkage matrix*. A detailed explanation of its format can be found in the SciPy documentation.

[2]:
```
from pyminimax import minimax
from scipy.spatial.distance import pdist

Z = minimax(pdist(X))
Z
```

[2]:
```
array([[ 0.        ,  1.        ,  1.        ,  2.        ],
       [ 2.        , 12.        ,  1.        ,  3.        ],
       [ 3.        ,  4.        ,  1.        ,  2.        ],
       [ 6.        ,  7.        ,  1.        ,  2.        ],
       [ 5.        , 14.        ,  1.        ,  3.        ],
       [ 8.        , 15.        ,  1.        ,  3.        ],
       [ 9.        , 10.        ,  1.        ,  2.        ],
       [11.        , 18.        ,  1.        ,  3.        ],
       [13.        , 16.        ,  3.16227766,  6.        ],
       [17.        , 19.        ,  3.16227766,  6.        ],
       [20.        , 21.        ,  5.        , 12.        ]])
```

Given the linkage matrix, one can then utilize the methods in SciPy to present the clustering result in a more readable manner. Below are examples applying `dendrogram` and `fcluster`.

[3]:
```
from scipy.cluster.hierarchy import dendrogram
from matplotlib import pyplot as plt

fig = plt.figure(figsize=(10, 4))
dendrogram(Z)
plt.show()
```




[4]:
```
from scipy.cluster.hierarchy import fcluster

fcluster(Z, t=1.8, criterion='distance')
```

[4]:
```
array([1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4], dtype=int32)
```

The above result says that, cutting the dendrogram at 1.8 threshold, the data has 4 clusters, with the first 3 points being in the first cluster, and the following 3 in the second cluster, and so on.

## 1.3 Getting Prototypes

A unique advantage of minimax-linkage hierarchical clustering is that in each cluster one prototype is selected from the original data. Thus the resulting clusters are better interpretable. Starting 0.1.0, PyMinimax can also compute those prototypes. Below we demonstrate how this can be done.

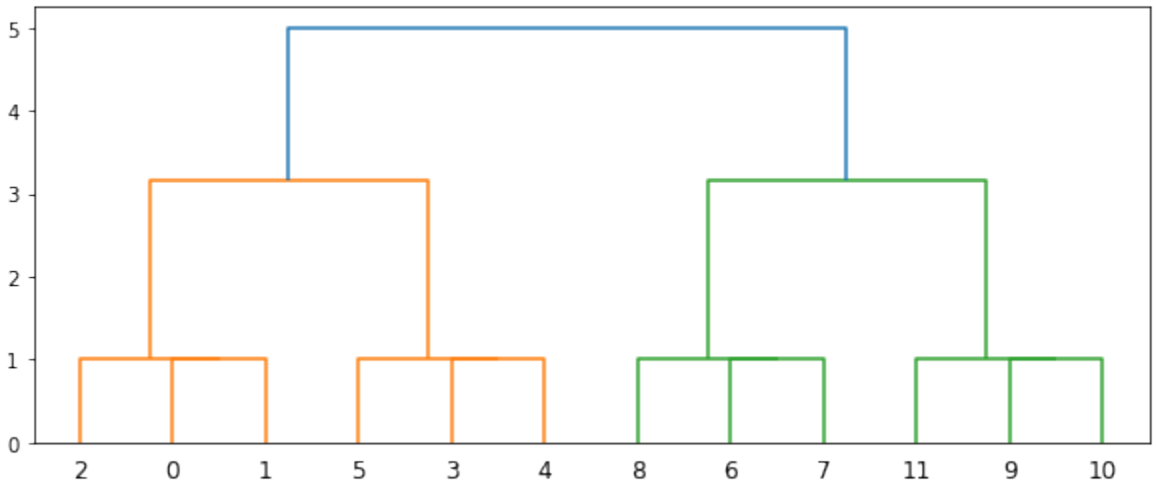
To obtain the prototypes, first we need an *extended linkage matrix* that contains prototypes information. This is an $(n-1)$ by 5 matrix, where the first 4 columns are in the same format as the standard linkage matrix and the 5th column keeps the indices of the prototypes of each merged cluster. The extended linkage matrix can be computed by `pyminimax.minimax` with `return_prototype=True`.

[5]:
```
Z_ext = minimax(pdist(X), return_prototype=True)
Z_ext
```

[5]:
```
array([[ 0.        ,  1.        ,  1.        ,  2.        ,  0.        ],
       [ 2.        , 12.        ,  1.        ,  3.        ,  0.        ],
       [ 3.        ,  4.        ,  1.        ,  2.        ,  3.        ],
       [ 6.        ,  7.        ,  1.        ,  2.        ,  6.        ],
       [ 5.        , 14.        ,  1.        ,  3.        ,  3.        ],
       [ 8.        , 15.        ,  1.        ,  3.        ,  6.        ],
       [ 9.        , 10.        ,  1.        ,  2.        ,  9.        ],
       [11.        , 18.        ,  1.        ,  3.        ,  9.        ],
       [13.        , 16.        ,  3.16227766,  6.        ,  1.        ],
       [17.        , 19.        ,  3.16227766,  6.        ,  8.        ],
       [20.        , 21.        ,  5.        , 12.        ,  1.        ]])
```

The 5-column extended linkage matrix is no longer SciPy-compatible. Any SciPy function that takes a linkage matrix will only work if we slice the extended linkage matrix to drop the 5th column, hence the `Z_ext[:, :4]` below.

[6]:
```
fig = plt.figure(figsize=(10, 4))
dendrogram(Z_ext[:, :4])
plt.show()
```




In PyMinimax, the prototypes are computed by `pyminimax.fcluster_prototype`. Its usage is exactly the same as `scipy.cluster.hierarchy.fcluster`, except

1. it takes the 5-column extended linkage matrix instead of a standard 4-column one, and
2. it returns information regarding clusters *and* prototypes.

[7]:
```
from pyminimax import fcluster_prototype

fcluster_prototype(Z_ext, t=1.8, criterion='distance')
```

[7]:
```
array([[1, 0],
       [1, 0],
       [1, 0],
       [2, 3],
       [2, 3],
       [2, 3],
       [3, 6],
       [3, 6],
       [3, 6],
       [4, 9],
       [4, 9],
       [4, 9]], dtype=int32)
```

The above result says that, cutting the dendrogram at 1.8 threshold, the data has 4 clusters. The first 3 points are in the first cluster and the prototype of this cluster is the 0th data point, the following 3 points are in the second cluster and the prototype of this cluster is the 3rd data point, and so on.

## 1.4 See Also

- scipy.cluster.hierarchy.complete
- scipy.cluster.hierarchy.dendrogram
- scipy.cluster.hierarchy.fcluster
- scipy.cluster.hierarchy.linkage
- scipy.spatial.distance.pdist

CHAPTER

TWO

# API REFERENCE

pyminimax.**minimax**(*dists*, *return_prototype=False*)

Perform minimax-linkage clustering using nearest-neighbor chain algorithm.

**Parameters**

- **dists** (*ndarray*) – The upper triangular of the distance matrix. The result of `scipy.spatial.distance.pdist` is returned in this form.
- **return_prototype** (*bool, default False*) – whether to return prototypes. When this is False, the returned linkage matrix Z has 4 columns, structured the same as the return value of the `scipy.cluster.hierarchy.linkage` function. When this is True, the returned linkage matrix has a 5th column which contains the indices of the prototypes corresponding to each merge.

**Returns** **Z** – A linkage matrix containing the hierarchical clustering. The first 4 columns has the same structure as the return value of the `scipy.cluster.hierarchy.linkage` function. See the documentation for more information on its structure. Depending on the value of `return_prototype` there is an optional 5th columns.

**Return type** ndarray

pyminimax.**fcluster_prototype**(*Z*, *t*, *criterion='inconsistent'*, *depth=2*, *R=None*, *monocrit=None*)

Form flat clusters from the hierarchical clustering defined by the given linkage matrix, and the

**Parameters**

- **Z** (*ndarray*) – The hierarchical clustering encoded with the matrix returned by the *minimax* function.
- **t** (*scalar*) –

  **For criteria 'inconsistent', 'distance' or 'monocrit',** this is the threshold to apply when forming flat clusters.

  **For 'maxclust' or 'maxclust_monocrit' criteria,** this would be max number of clusters requested.
- **criterion** (*str, optional*) – The criterion to use in forming flat clusters. This can be any of the following values:

  **inconsistent :** If a cluster node and all its descendants have an inconsistent value less than or equal to *t*, then all its leaf descendants belong to the same flat cluster. When no non-singleton cluster meets this criterion, every node is assigned to its own cluster. (Default)

  **distance :** Forms flat clusters so that the original observations in each flat cluster have no greater a cophenetic distance than *t*.

**maxclust :** Finds a minimum threshold `r` so that the cophenetic distance between any two original observations in the same flat cluster is no more than `r` and no more than *t* flat clusters are formed.

**monocrit :** Forms a flat cluster from a cluster node c with index i when `monocrit[j] <= t`. For example, to threshold on the maximum mean distance as computed in the inconsistency matrix R with a threshold of 0.8 do:

```
MR = maxRstat(Z, R, 3)
fcluster_prototype(Z, t=0.8, criterion='monocrit', monocrit=MR)
```

**maxclust_monocrit :** Forms a flat cluster from a non-singleton cluster node `c` when `monocrit[i] <= r` for all cluster indices `i` below and including `c`. `r` is minimized such that no more than `t` flat clusters are formed. monocrit must be monotonic. For example, to minimize the threshold t on maximum inconsistency values so that no more than 3 flat clusters are formed, do:

```
MI = maxinconsts(Z, R)
fcluster_prototype(Z, t=3, criterion='maxclust_monocrit',␣
↪monocrit=MI)
```

- **depth** (*int, optional*) – The maximum depth to perform the inconsistency calculation. It has no meaning for the other criteria. Default is 2.
- **R** (*ndarray, optional*) – The inconsistency matrix to use for the 'inconsistent' criterion. This matrix is computed if not provided.
- **monocrit** (*ndarray, optional*) – An array of length n-1. *monocrit[i]* is the statistics upon which non-singleton i is thresholded. The monocrit vector must be monotonic, i.e., given a node c with index i, for all node indices j corresponding to nodes below c, `monocrit[i] >= monocrit[j]`.

**Returns** **fcluster_prototype** – An array of shape `(n, 2)`. `T[i]` is the flat cluster number to which original observation `i` belongs, and the index of the prototype of this cluster.

**Return type** ndarray

CHAPTER

THREE

# REFERENCE

# INDEX

## F



## M