pydist2 guide

Whether you need to compute the distances between observation of vectors, `distance` does it all!

Special Distances

There are several kinds of distance for which `distance` offers special support.

A. **pdist1** calculates the pairwise distances between observations in one vector with one of the following methods:

For a given a matrix of points $X_{m,n}$ (m rows and n columns), where each row is treated as a vector of points X_{1i} = $(x_{1i}, x_{2i}, \ldots, x_{ni})$, the various distances between the vector X_i and X_j are defined as follows:

1. **euclidean**: $d_{X_j,X_k} = \sqrt{\sum_{i=1}^{n}(x_{ij} - x_{ik})^2}$
2. **default**: the default method is the euclidean distance.
3. **seuclidean**: standardized euclidean distance. $d_{X_j,X_k} = \sqrt{\sum_{i=1}^{n} w_i(x_{ij} - x_{ik})^2}$ where w represents the inverse of the variance of the vector Xj over the m vectors.
4. **cityblock**: $d_{X_j,X_k} = \sum_{i=1}^{n} |x_{ij} - x_{ik}|$
5. **mahalanobis**: $d_{X_j,X_k} = \sum_{i=1}^{n} V_i(x_{ij} - x_{ik})^2$ where V is the inverse of the covariance matrix of X.
6. **minkowski**: $d_{X_j,X_k} = (\sum_{i=1}^{n} |x_{ij} - x_{ik})^p|^{1/p}$
7. **chebyshev**: $d_{X_j,X_k} = max|X_j - X_k|$
8. **cosine**: $d_{X_j,X_k} = 1 - cos(\boldsymbol{X}_j, \boldsymbol{X}_k) = 1 - \frac{\boldsymbol{X}_j \cdot \boldsymbol{X}_k}{||\boldsymbol{X}_j|| \cdot ||\boldsymbol{X}_k||} = 1 - \frac{\sum_{i=1}^{n} \boldsymbol{x}_{ij} \cdot \boldsymbol{x}_{ik}}{\sqrt{\sum_{i=1}^{n} x_{ij}^2} \cdot \sqrt{\sum_{i=1}^{n} x_{ik}^2}}$
9. **correlation**: $d_{X_j,X_k} = 1 - \frac{\sum_{i=1}^{n}(x_{ij} - (1/n) \cdot \sum_{j=1}^{n} x_{ij}) \cdot (x_{ik} - (1/n) \cdot \sum_{k=1}^{n} x_{ik})}{\sqrt{\sum_{i=1}^{n}(x_{ij} - (1/n) \cdot \sum_{j=1}^{n} x_{ij})^2} \cdot \sqrt{\sum_{i=1}^{n}(x_{ik} - (1/n) \cdot \sum_{k=1}^{n} x_{ik})^2}}$
10. **spearman**: $d_{X_j,X_k} = 1 - \frac{\sum_{i=1}^{n}(r_{ij} - \frac{(n+1)}{2}) \cdot (r_{ik} - \frac{(n+1)}{2})}{\sqrt{\sum_{i=1}^{n}(r_{ij} - \frac{(n+1)}{2})^2} \cdot \sqrt{\sum_{i=1}^{n}(r_{ik} - \frac{(n+1)}{2})^2}}$
11. **hamming**: $d_{X_j,X_k} = \frac{||(X_j \otimes X_k) \cap mask_{X_j} \cap mask_{X_k}||}{||mask_{X_j} \cap mask_{X_k}||}$
12. **jaccard**: $d_{X_j,X_k} = \frac{|X_j \bigcap X_k|}{|X_j \bigcup X_k|}$

B. **pdist2** calculates the distances between observations in two vectors with one of the following methods:

1. **manhattan**: The L1 distance between two vectors P and Q is defined as: $d(P,Q) = ||P - Q||_1 = \sum_{i=1}^{n} |p_i - q_i|$
2. **sqeuclidean**: Euclidean squared distance defined as: $d(P,Q)^2 = \sum_{i=1}^{n}(p_i - q_i)^2$
3. **euclidean**: Euclidean distance defined as: $d(P,Q) = \sqrt{\sum_{i=1}^{n}(P-Q)^2}$
4. **default**: the default method is the euclidean distance.
5. **chi-squared**: $d_{P,Q} = \sum_{i=1}^{n} \frac{(P_i - Q_i)^2}{P_i + Q_i}$
6. **cosine**:
$$1 - Cosine\_Similarity = 1 - cos(\boldsymbol{P}, \boldsymbol{Q}) = 1 - \frac{\boldsymbol{P} \cdot \boldsymbol{Q}}{||\boldsymbol{P}|| \cdot ||\boldsymbol{Q}||} = 1 - \frac{\sum_{i=1}^{n} \boldsymbol{P}_i \cdot \boldsymbol{Q}_i}{\sqrt{\sum_{i=1}^{n} P_i^2} \cdot \sqrt{\sum_{i=1}^{n} Q_i^2}}$$
7. **earthmover**: $EMD(P,Q) = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n} f_{ij} d_{ij}}{\sum_{i=1}^{m}\sum_{j=1}^{b} f_{ij}}$

These are supported by simple classes that are available in the `distance` module, and also by a pair of classes of the main `pdist1` and `pdist2` classes.

pdist1 and pdist2

The library can compute distances between pair of observations in one vector using `pdist1`, and distances between pair of observations in two vectors using `pdist2`. Note that the two vectors must have the same shape!

```python
>>> from pydist2.distance import pdist1, pdist2
>>> import numpy as np
>>> x = np.array([[1, 2, 3],
...         [7, 8, 9],
...         [5, 6, 7],], dtype=np.float32)
>>> y = np.array([[10, 20, 30],
...         [70, 80, 90],
...         [50, 60, 70]], dtype=np.float32)
>>> a = pdist1(x)
>>> a
array([10.39230485,  6.92820323,  3.46410162])
>>> pdist1(x, 'seuclidean')
array([3.40168018, 2.26778677, 1.13389339])
>>> pdist1(x, 'minkowski', exp=3)
array([8.65349742, 5.76899828, 2.88449914])
>>> pdist1(x, 'minkowski', exp=2)
array([10.39230485,  6.92820323,  3.46410162])
>>> pdist1(x, 'minkowski', exp=1)
array([18., 12.,  6.])
>>> pdist1(x, 'cityblock')
array([18., 12.,  6.])
>>> pdist2(x, y)
array([[ 33.67491648, 135.69819453, 101.26203632],
       [ 24.37211521, 125.35549449,  90.96153033],
       [ 27.38612788, 128.80217389,  94.39279634]])
>>> pdist2(x, y, 'manhattan')
array([[ 54., 234., 174.],
       [ 36., 216., 156.],
       [ 42., 222., 162.]])
>>> pdist2(x, y, 'sqeuclidean')
array([[ 1134., 18414., 10254.],
       [  594., 15714.,  8274.],
       [  750., 16590.,  8910.]])
>>> pdist2(x, y, 'chi-squared')
array([[ 22.09090909, 111.31927838,  81.41482329],
       [  8.48998061,  88.36363636,  59.6522841 ],
       [ 11.75121275,  95.51418525,  66.27272727]])
>>> pdist2(x, y, 'cosine')
array([[-5.60424152e-09,  4.05881305e-02,  3.16703408e-02],
       [ 4.05880431e-02,  7.31070616e-08,  5.62480978e-04],
       [ 3.16703143e-02,  5.62544701e-04, -1.23279462e-08]])
>>> pdist2(x, y, 'earthmover')
array([[ 90., 450., 330.],
       [ 54., 414., 294.],
       [ 66., 426., 306.]])
```