6.8. Pairwise metrics, Affinities and Kernels

The `sklearn.metrics.pairwise` submodule implements utilities to evaluate pairwise distances or affinity of sets of samples.

This module contains both distance metrics and kernels. A brief summary is given on the two here.

Distance metrics are functions `d(a, b)` such that `d(a, b) < d(a, c)` if objects `a` and `b` are considered "more similar" than objects `a` and `c`. Two objects exactly alike would have a distance of zero. One of the most popular examples is Euclidean distance. To be a 'true' metric, it must obey the following four conditions:

```
1. d(a, b) >= 0, for all a and b
2. d(a, b) == 0, if and only if a = b, positive definiteness
3. d(a, b) == d(b, a), symmetry
4. d(a, c) <= d(a, b) + d(b, c), the triangle inequality
```

Kernels are measures of similarity, i.e. `s(a, b) > s(a, c)` if objects `a` and `b` are considered "more similar" than objects `a` and `c`. A kernel must also be positive semi-definite.

There are a number of ways to convert between a distance metric and a similarity measure, such as a kernel. Let `D` be the distance, and `S` be the kernel:

1. `S = np.exp(-D * gamma)`, where one heuristic for choosing `gamma` is `1 / num_features`
2. `S = 1. / (D / np.max(D))`

The distances between the row vectors of `X` and the row vectors of `Y` can be evaluated using `pairwise_distances`. If `Y` is omitted the pairwise distances of the row vectors of `X` are calculated. Similarly, `pairwise.pairwise_kernels` can be used to calculate the kernel between `X` and `Y` using different kernel functions. See the API reference for more details.

```
>>> import numpy as np
>>> from sklearn.metrics import pairwise_distances
>>> from sklearn.metrics.pairwise import pairwise_kernels
>>> X = np.array([[2, 3], [3, 5], [5, 8]])
>>> Y = np.array([[1, 0], [2, 1]])
>>> pairwise_distances(X, Y, metric='manhattan')
array([[ 4.,  2.],
       [ 7.,  5.],
       [12., 10.]])
>>> pairwise_distances(X, metric='manhattan')
array([[0., 3., 8.],
       [3., 0., 5.],
       [8., 5., 0.]])
>>> pairwise_kernels(X, Y, metric='linear')
array([[ 2.,  7.],
       [ 3., 11.],
       [ 5., 18.]])
```

6.8.1. Cosine similarity

`cosine_similarity` computes the L2-normalized dot product of vectors. That is, if x and y are row vectors, their cosine similarity k is defined as:

$$k(x, y) = \frac{x y^\top}{\|x\| \|y\|}$$

This is called cosine similarity, because Euclidean (L2) normalization projects the vectors onto the unit sphere, and their dot product is then the cosine of the angle between the points denoted by the vectors.

This kernel is a popular choice for computing the similarity of documents represented as tf-idf vectors. `cosine_similarity` accepts `scipy.sparse` matrices. (Note that the tf-idf functionality in `sklearn.feature_extraction.text` can produce normalized vectors, in which case `cosine_similarity` is equivalent to `linear_kernel`, only slower.)

References:

- C.D. Manning, P. Raghavan and H. Schütze (2008). Introduction to Information Retrieval. Cambridge University Press. https://nlp.stanford.edu/IR-book/html/htmledition/the-vector-space-model-for-scoring-1.html

Toggle Menu

6.8.2. Linear kernel

The function `linear_kernel` computes the linear kernel, that is, a special case of `polynomial_kernel` with `degree=1` and `coef0=0` (homogeneous). If `x` and `y` are column vectors, their linear kernel is:

$$k(x, y) = x^\top y$$

6.8.3. Polynomial kernel

The function `polynomial_kernel` computes the degree-d polynomial kernel between two vectors. The polynomial kernel represents the similarity between two vectors. Conceptually, the polynomial kernels considers not only the similarity between vectors under the same dimension, but also across dimensions. When used in machine learning algorithms, this allows to account for feature interaction.

The polynomial kernel is defined as:

$$k(x, y) = (\gamma x^\top y + c_0)^d$$

where:

- `x`, `y` are the input vectors
- `d` is the kernel degree

If $c_0 = 0$ the kernel is said to be homogeneous.

6.8.4. Sigmoid kernel

The function `sigmoid_kernel` computes the sigmoid kernel between two vectors. The sigmoid kernel is also known as hyperbolic tangent, or Multilayer Perceptron (because, in the neural network field, it is often used as neuron activation function). It is defined as:

$$k(x, y) = \tanh(\gamma x^\top y + c_0)$$

where:

- `x`, `y` are the input vectors
- γ is known as slope
- c_0 is known as intercept

6.8.5. RBF kernel

The function `rbf_kernel` computes the radial basis function (RBF) kernel between two vectors. This kernel is defined as:

$$k(x, y) = \exp(-\gamma \|x - y\|^2)$$

where `x` and `y` are the input vectors. If $\gamma = \sigma^{-2}$ the kernel is known as the Gaussian kernel of variance σ^2.

6.8.6. Laplacian kernel

The function `laplacian_kernel` is a variant on the radial basis function kernel defined as:

$$k(x, y) = \exp(-\gamma \|x - y\|_1)$$

where `x` and `y` are the input vectors and $\|x - y\|_1$ is the Manhattan distance between the input vectors.

It has proven useful in ML applied to noiseless data. See e.g. Machine learning for quantum mechanics in a nutshell.

6.8.7. Chi-squared kernel

The chi-squared kernel is a very popular choice for training non-linear SVMs in computer vision applications. It can be computed using `chi2_kernel` and then passed to an `SVC` with `kernel="precomputed"`:

Toggle Menu

```
>>> from sklearn.svm import SVC
>>> from sklearn.metrics.pairwise import chi2_kernel
>>> X = [[0, 1], [1, 0], [.2, .8], [.7, .3]]
>>> y = [0, 1, 0, 1]
>>> K = chi2_kernel(X, gamma=.5)
>>> K
array([[1.        , 0.36787944, 0.89483932, 0.58364548],
       [0.36787944, 1.        , 0.51341712, 0.83822343],
       [0.89483932, 0.51341712, 1.        , 0.7768366 ],
       [0.58364548, 0.83822343, 0.7768366 , 1.        ]])

>>> svm = SVC(kernel='precomputed').fit(K, y)
>>> svm.predict(K)
array([0, 1, 0, 1])
```

It can also be directly used as the `kernel` argument:

```
>>> svm = SVC(kernel=chi2_kernel).fit(X, y)
>>> svm.predict(X)
array([0, 1, 0, 1])
```

The chi squared kernel is given by

$$k(x, y) = \exp \left(-\gamma \sum_i \frac{(x[i] - y[i])^2}{x[i] + y[i]} \right)$$

The data is assumed to be non-negative, and is often normalized to have an L1-norm of one. The normalization is rationalized with the connection to the chi squared distance, which is a distance between discrete probability distributions.

The chi squared kernel is most commonly used on histograms (bags) of visual words.

References:

- Zhang, J. and Marszalek, M. and Lazebnik, S. and Schmid, C. Local features and kernels for classification of texture and object categories: A comprehensive study International Journal of Computer Vision 2007 https://hal.archives-ouvertes.fr/hal-00171412/document

Toggle Menu