

10個機器學習中常用的距離度量方法

機器學習算法與Python實戰 2022-11-23 08:00 發表於奧地利

收錄於合集

#機器學習 74 #算法 6 #無監督學習 1



機器學習算法與Python實戰

長期跟踪關注統計學、數據挖掘、機器學習算法、深度學習、人工智能技術與行業發展...

294篇原創內容

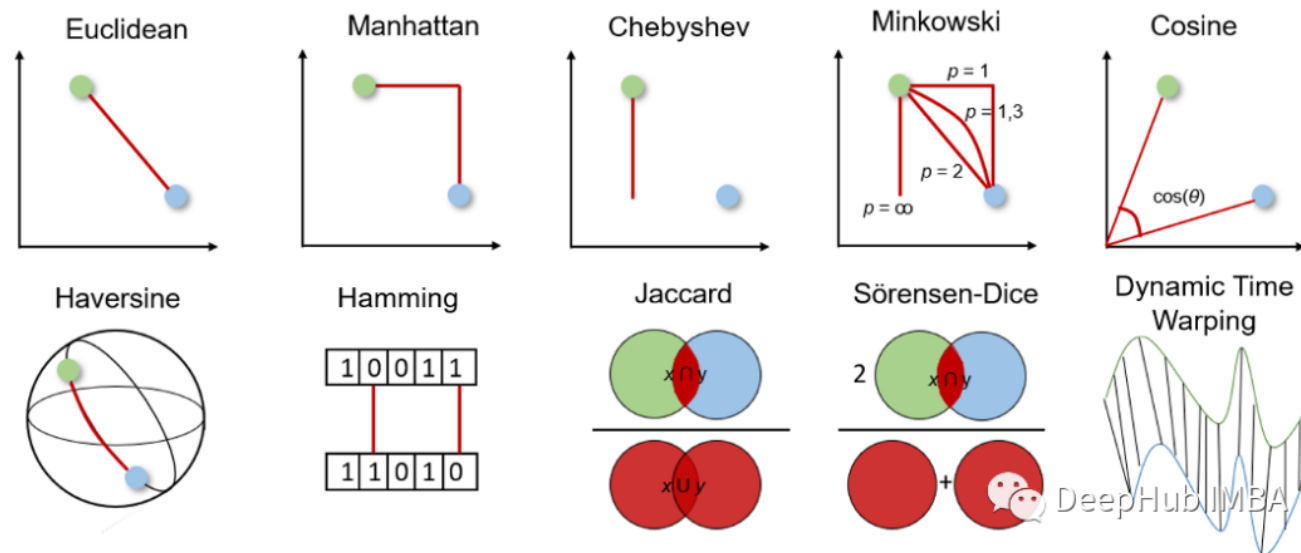
公眾號

轉載：Deephub Imba

距離度量是有監督和無監督學習算法的基礎，包括k近鄰、支持向量機和k均值聚類等。

距離度量的選擇影響我們的機器學習結果，因此考慮哪種度量最適合這個問題是很重要的。因此，我們在決定使用哪種測量方法時應該謹慎。但在做出決定之前，我們需要了解距離測量是如何工作的，以及我們可以從哪些測量中進行選擇。

本文將簡要介紹常用的距離度量方法、它們的工作原理、如何用Python計算它們以及何時使用它們。這樣可以加深知識和理解，提高機器學習算法和結果。



在更深入地研究不同的距離測量之前，我們先要有一個關於它們如何工作以及如何選擇合適的測量的大致概念。

距離度量用於計算給定問題空間中兩個對象之間的差異，即數據集中的特徵。然後可以使用該距離來確定特徵之間的相似性，距離越小特徵越相似。

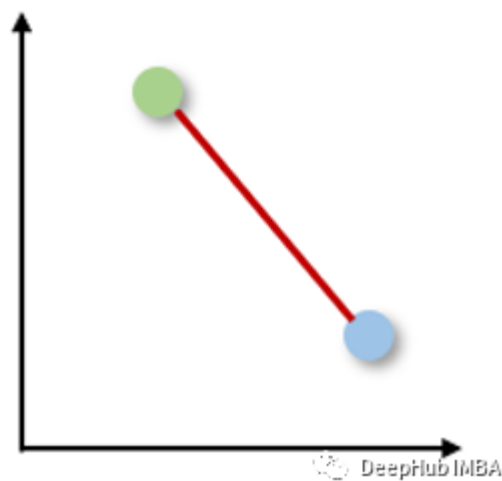
對於距離的度量，我們可以在幾何距離測量和統計距離測量之間進行選擇，應該選擇哪種距離度量取決於數據的類型。特徵可能有不同的數據類型（例如，真實值、布爾值、分類值），數據可能是多維的或由地理空間數據組成。

幾何距離測量

1、歐氏距離Euclidean distance

歐氏距離度量兩個實值向量之間的最短距離。由於其直觀，使用簡單和對許多用例有良好結果，所以它是最常用的距離度量和許多應用程序的默認距離度量。

Euclidean



歐氏距離也可稱為L2範數，其計算方法為：

$$d = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

A small logo with the text 'DeepHub IMBA' is visible in the bottom right corner of the equation block.

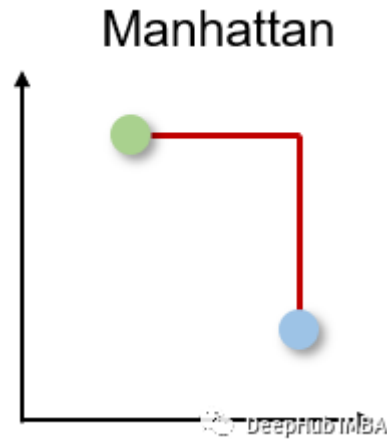
Python代碼如下

```
from scipy.spatial import distance  
distance.euclidean(vector_1, vector_2)
```

歐氏距離有兩個主要缺點。首先，距離測量不適用於比2D或3D空間更高維度的數據。第二，如果我們不將特徵規範化和/或標準化，距離可能會因為單位的不同而傾斜。

2、曼哈頓距離Manhattan distance

曼哈頓距離也被稱為出租車或城市街區距離，因為兩個實值向量之間的距離是根據一個人只能以直角移動計算的。這種距離度量通常用於離散和二元屬性，這樣可以獲得真實的路徑。



曼哈頓距離以L1範數為基礎，計算公式為：

$$d = \sum_{i=1}^n (x_i - y_i)$$

DeepHubIMBA

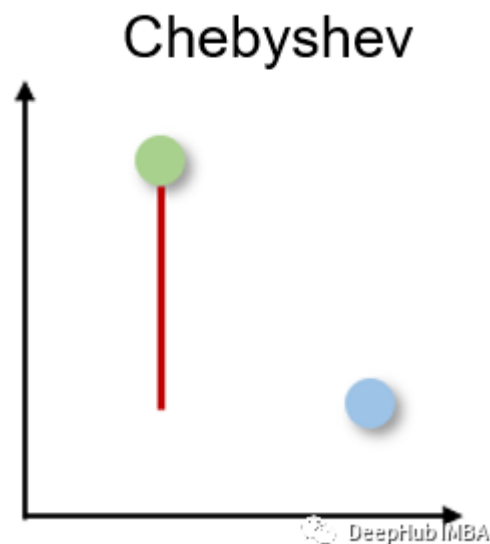
Python代碼如下

```
from scipy.spatial import distance  
distance.cityblock(vector_1, vector_2)
```

曼哈頓的距離有兩個主要的缺點。它不如高維空間中的歐氏距離直觀，它也沒有顯示可能的最短路徑。雖然這可能沒有問題，但我們應該意識到這並不是最短的距離。

3、切比雪夫距離Chebyshev distance

切比雪夫距離也稱為棋盤距離，因為它是兩個實值向量之間任意維度上的最大距離。它通常用於倉庫物流中，其中最長的路徑決定了從一個點到另一個點所需的時間。



切比雪夫距離由l -無窮範數計算:

$$d = \max_i (|x_i - y_i|)$$

DeepHub IMBA

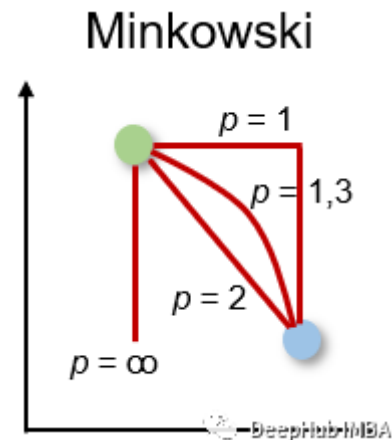
Python代碼如下

```
from scipy.spatial import distance
distance.chebyshev(vector_1, vector_2)
```

切比雪夫距離只有非常特定的用例，因此很少使用。

4、閔可夫斯基距離Minkowski distance

閔可夫斯基距離是上述距離度量的廣義形式。它可以用於相同的用例，同時提供高靈活性。我們可以選擇 p 值來找到最合適的距離度量。



閔可夫斯基距離的計算方法為:

$$d = \sqrt[p]{\sum_{i=1}^n (x_i - y_i)^p}$$

DeepHub IMBA

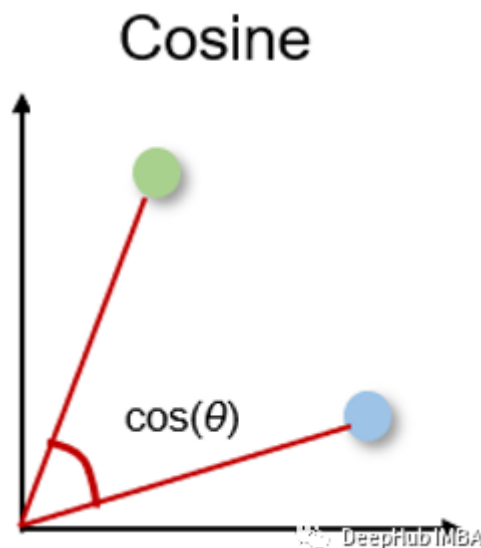
Python代碼如下

```
from scipy.spatial import distance  
distance.minkowski(vector_1, vector_2, p)
```


由於閔可夫斯基距離表示不同的距離度量，它就有與它們相同的主要缺點，例如在高維空間的問題和對特徵單位的依賴。此外， p 值的靈活性也可能是一個缺點，因為它可能降低計算效率，因為找到正確的 p 值需要進行多次計算。

5、餘弦相似度和距離Cosine similarity

餘弦相似度是方向的度量，他的大小由兩個向量之間的餘弦決定，並且忽略了向量的大小。餘弦相似度通常用於與數據大小無關緊要的高維，例如，推薦系統或文本分析。



餘弦相似度可以介於-1(相反方向)和1(相同方向)之間，計算方法為:

$$d = \cos(\alpha) = \frac{x y}{\|x\| \|y\|}$$


餘弦相似度常用於範圍在0到1之間的正空間中。餘弦距離就是用1減去餘弦相似度，位於0(相似值)和1(不同值)之間。

Python代碼如下

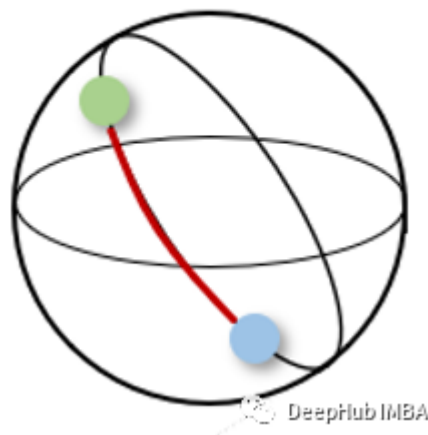
```
from scipy.spatial import distance  
distance.cosine(vector_1, vector_2)
```

餘弦距離的主要缺點是它不考慮大小而只考慮向量的方向。因此，沒有充分考慮到值的差異。

6、半正矢距離Haversine distance

半正矢距離測量的是球面上兩點之間的最短距離。因此常用於導航，其中經度和緯度和曲率對計算都有影響。

Haversine



半正矢距離的公式如下：

$$d = 2r \arcsin \left(\sqrt{\sin^2 \left(\frac{\varphi_2 - \varphi_1}{2} \right) + \cos \varphi_1 \cos \varphi_2 \sin^2 \left(\frac{\lambda_2 - \lambda_1}{2} \right)} \right)$$

其中 r 為球面半徑， φ 和 λ 為經度和緯度。

Python代碼如下

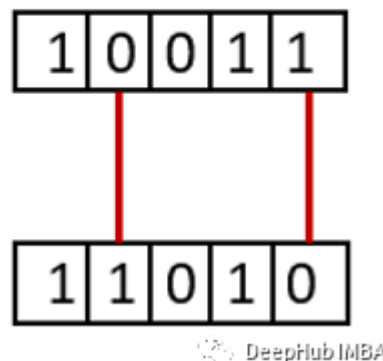
```
from sklearn.metrics.pairwise import haversine_distances
haversine_distances([vector_1, vector_2])
```

半正矢距離的主要缺點是假設是一個球體，而這種情況很少出現。

7、漢明距離

漢明距離衡量兩個二進制向量或字符串之間的差異。

Hamming



對向量按元素進行比較，並對差異的數量進行平均。如果兩個向量相同，得到的距離是0之間，如果兩個向量完全不同，得到的距離是1。

Python代碼如下

```
from scipy.spatial import distance  
distance.hamming(vector_1, vector_2)
```

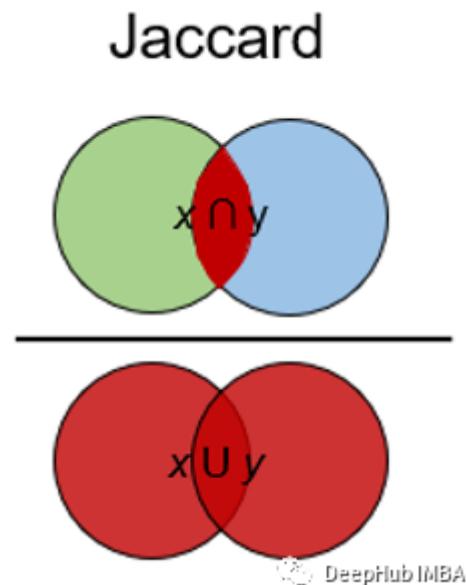
漢明距離有兩個主要缺點。距離測量只能比較相同長度的向量，它不能給出差異的大小。所以當差異的大小很重要時，不建議使用漢明距離。

統計距離測量

統計距離測量可用於假設檢驗、擬合優度檢驗、分類任務或異常值檢測。

8、杰卡德指數和距離Jaccard Index

Jaccard指數用於確定兩個樣本集之間的相似性。它反映了與整個數據集相比存在多少一對一匹配。Jaccard指數通常用於二進制數據比如圖像識別的深度學習模型的預測與標記數據進行比較，或者根據單詞的重疊來比較文檔中的文本模式。



Jaccard距離的計算方法為:

$$d = 1 - \frac{|x \cap y|}{|x \cup y|}$$

DeepHub IMBA

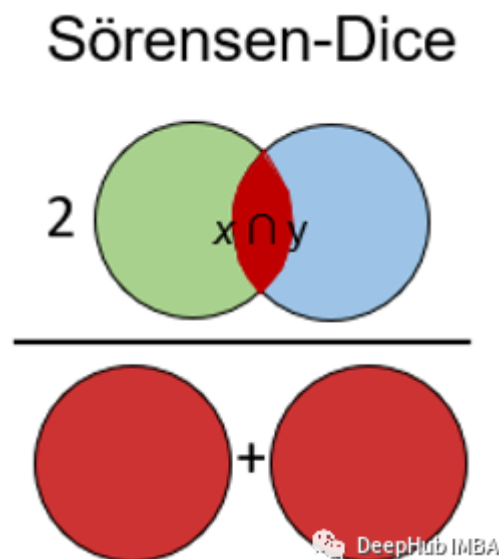
Python代碼如下

```
from scipy.spatial import distance  
distance.jaccard(vector_1, vector_2)
```


Jaccard指數和距離的主要缺點是，它受到數據規模的強烈影響，即每個項目的權重與數據集的規模成反比。

9、Sorensen-Dice指數

Sørensen-Dice指數類似於Jaccard指數，它可以衡量的是樣本集的相似性和多樣性。該指數更直觀，因為它計算重疊的百分比。Sørensen-Dice索引常用於圖像分割和文本相似度分析。



計算公式如下：

$$d = \frac{2|x \cap y|}{|x| + |y|}$$


Python代碼如下

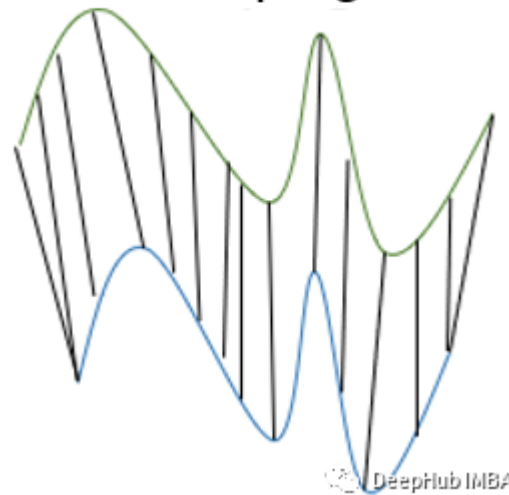
```
from scipy.spatial import distance  
distance.dice(vector_1, vector_2)
```

它的主要缺點也是受數據集大小的影響很大。

10、動態時間規整Dynamic Time Warping

動態時間規整是測量兩個不同長度時間序列之間距離的一種重要方法。可以用於所有時間序列數據的用例，如語音識別或異常檢測。

Dynamic Time Warping



為什麼我們需要一個為時間序列進行距離測量的度量呢？如果時間序列長度不同或失真，則上述面說到的其他距離測量無法確定良好的相似性。比如歐幾里得距離計算每個時間步長的兩個時間序列之間的距離。但是如果兩個時間序列的形狀相同但在時間上發生了偏移，那麼儘管時間序列非常相似，但歐幾里得距離會表現出很大的差異。

動態時間規整通過使用多對一或一對多映射來最小化兩個時間序列之間的總距離來避免這個問題。當搜索最佳對齊時，這會產生更直觀的相似性度量。通過動態規劃找到一條彎曲的路徑最小化距離，該路徑必須滿足以下條件：

邊界條件:彎曲路徑在兩個時間序列的起始點和結束點開始和結束

單調性條件:保持點的時間順序，避免時間倒流

連續條件:路徑轉換限制在相鄰的時間點上，避免時間跳躍

整經窗口條件(可選):允許的點落入給定寬度的整經窗口

坡度條件(可選):限制彎曲路徑坡度，避免極端運動

我們可以使用Python 中的fastdtw 包：

```
from scipy.spatial.distance import euclidean
from fastdtw import fastdtw

distance, path = fastdtw(timeseries_1, timeseries_2, dist=euclidean)
```

動態時間規整的一個主要缺點是與其他距離測量方法相比，它的計算工作量相對較高。

總結

在這篇文章中，簡要介紹了十種常用的距離測量方法。本文中已經展示了它們是如何工作的，如何在Python中實現它們，以及經常使用它們解決什麼問題。如果你認為我錯過了一個重要的距離測量，請留言告訴我。

作者：Jonte Dancker



推薦閱讀

全網最全速查表：Python 機器學習

搭建完美的Python 機器學習開發環境

訓練集，驗證集，測試集，交叉驗證

Matplotlib 三連彈

收錄於合集#機器學習 74

下一篇 · 人人都能看懂的EM算法推導

喜歡此內容的人還喜歡

20天吃透PyTorch (附下載)

Python數據科學



Python 3.11正式版來了，比3.10快10-60%，官方：這或許是最好的版本

Python數據科學



機器學習模型迭代方法(Python)

算法進階

