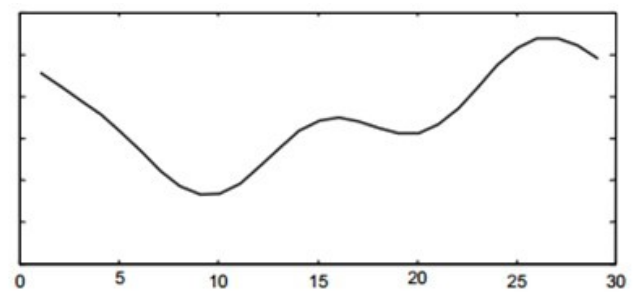
时间序列那点事儿

分类篇（二）

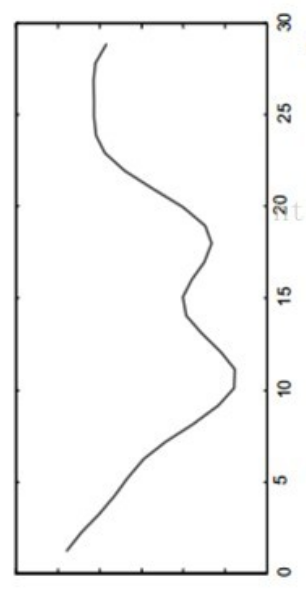
**DTW (**Dynamic Time Warping,动态时间归整)

**什么是DTW?**

DTW算法是一种衡量两种长短不一的时间序列相似度的方法



当前两个序列是否相似？这样的相似，应该如何去量化？



DTW

**为什么时间序列分类要用到DTW?**

对于基于距离分类的分类算法，我们知道关键点是解决如何根据特征将每个样本投影在空间中或者是如何比较两个样本之间的距离？

这样一来我们就能根据距离当前测试样本相聚最近的K个训练样本点来决定当前测试样本的类别。所以我们需要一个方法去量化地衡量两个时间序列样本的相似性

A类

B类

C类

距离：10

距离：5

距离：3

C类

DTW 作为一种衡量两个时间序列相似的，对于其他衡量时间相似度的方法（欧式距离）相比可以一定程度上忽略时间序列的压缩和拉升来衡量两个时间序列是否相似。

理论上，对于一对时间序列，用欧式距离衡量相似度的方法如下：

（L2\_t1 - L1\_t1）.^2

L1\_t1

L2\_t1

所以我们知道，衡量L1，L2相似的条件是，计算L1，L2两个序列每一个时间点取值的相差的大小（这里用欧式距离来衡量）

在时间序列中，需要比较相似性的两段时间序列的长度可能并不相等，在语音识别领域表现为不同人的语速不同。因为语音信号具有相当大的随机性，即使同一个人在不同时刻发同一个音，也不可能具有完全的时间长度。而且同一个单词内的不同音素的发音速度也不同，比如有的人会把“A”这个音拖得很长，或者把“i”发的很短。在这些复杂情况下，使用传统的欧几里得距离无法有效地求的两个时间序列之间的距离（或者相似性）。

Key point : 采样

**DTW流程：**

Step1:

构建距离矩阵d,矩阵的内容是两个时间序列的每个时间的取值的距离

序列L1：l1\_t1 , l1\_t2 , … ,l1\_tn

序列L2：l2\_t1 , l2\_t2 , … ,l2\_tm

Sum(x,y) = (x-y).^2

d =

L2

L1

Step2:

构建路径矩阵D，该矩阵和距离矩阵d大小相同，但里面的元素都取INF。然后根据三个约束更新矩阵D，更新完毕后矩阵D的最后一个元素D[m][n] 就是两个序列的距离。**D矩阵每个元素代表的是对比到当前位置的累积距离**

D = (m x n)

不断更新D

序列L1：v1,v2,…,vm

序列L2：

v1,

v2,

…,

vn

D = (m x n)

**Distance = vnm**

而具体的更新规则如下：

根据距离矩阵d动态求解两个时间序列的最短差距，本质上，这个算法是要比较长短不一的两个时间序列，该算法需要满足三个约束：

1. **边界条件：**D=(1, 1) ( **D(1,1)=d(1,1)** ) 和D=(m, n)。任何一种语音的发音快慢都有可能变化，但是其各部分的先后次序不可能改变，因此所选的路径必定是从矩阵的左上角出发，在矩阵右下角结束。

更新起始点

D = (m x n)

更新终止点

**2）连续性：**假设，根据先后顺序将需要比较的点对用w来表示， =（a,b）表示第k次比较是比较序列L1 第a个点和序列L2第b个点。则如果= (a’, b’)，那么对于路径的下一个点=(a, b)需要满足 (a-a’) <=1和 (b-b’) <=1。也就是不可能跨过某个点去匹配，只能和自己相邻的点对齐。这样可以保证序列L1和序列L2中的每个坐标都在W中出现。

**3）单调性：**如果= (a’, b’)，那么对于路径的下一个点=(a, b)需要满足0<=(a-a’)和0<= (b-b’)。这限制W上面的点必须是随着时间单调进行的。以保证图B中的虚线不会相交。

由条件2,3可知，每次更新D的时候只有三个方向可以选择

D = (m x n)

inf inf

inf inf

注：连续性决定了更新的步长为1，不能跨步更新，确保尽可能多的点被比较

单调性决定了更新的方向只能 从左到右 或者 从上到下

**DTW详解：**

对于DTW的原理和算法流程我们大致掌握了，但是在具体编程上还有些细节值得注意的。

**首先D（1,1）= d(1,1)**

初始化：D =

**其次，每次更新D都是根据d矩阵和当前已有的D矩阵来更新的**。

D =

每个位置都有可能由三个方向的数据加上当前位置的d更新而来，

D = (m x n)

= d(2,2) + min(,,)



**最后D矩阵每个元素代表的是对比到当前位置的累积距离。**

序列L1：v1,v2,…,vm

序列L2：

v1,

v2,

…,

vn

D = (m x n)