班　级　0712**

学　号　0712****

西安电子科技大学

本科毕业设计论文

题　　目　基于 XDU*thesis* 模板的论文写作样例

学　　院　数学与统计学院

专　　业　统计学

学生姓名　刘精昌

导师姓名　崔元顺

毕业设计（论文）诚信声明书

本人声明：本人所提交的毕业论文《科学数据的序列分类》是本人在指导教师指导下独立研究、写作成果，论文中所引用他人的无论以何种方式发布的文字、研究成果，均在论文中加以说明；有关教师、同学和其他人员对本文本的写作、修订提出过并为我在论文中加以采纳的意见、建议，均已在我的致谢辞中加以说明并深致谢意。

本文和资料若有不实之处，本人承担一切相关责任。

论文作者：_____（签字）　时间：

指导教师已阅：_____（签字）　时间：

摘要

本测试样例，供 XDU*thesis* 模板使用者学习之用，会尽量将模板涉及到的命令、环境；以及一些常用的图表、公式排版技巧进行展示，会对参考文献进行进一步阐述。

其余不太重要的部分，会以其他内容进行展示，例如，诗歌、歌词等。

关键词：XDUthesis　**命令**　**环境**　**排版技巧**

Abstract

This paper is just a sample example for the users in learning the XDU*thesis*. I will try my best to use the commands and environments which are involved by the XDU*thesis*. Also, the popular composition skills in figures, tables and equations will be elaborated.

In the part unimportant, I will show something others, such as poems and lyrics.

Key Words: XDUthesis commands environments skills

目录

第一章 绪论

1.1 序列以及序列分类

现实生活中会有各种各样的序列。在语音识别领域，一段语音可以被看作一段序列；在经济领域，股票的走势可以被看作是序列图示；在分子生物学领域，蛋白质、DNA 也可以看成是一段序列；在网络管理中，客户端的登录活动也可以被记录成一段序列。而序列分类在现实生活中也有着广泛的应用。以语音库中的不同样本的语音为训练集，通过对需要测试的语音进行分类，可以识别出语音的来源；在医疗领域，通过对心电图序列的分类，可以得到这个疑似患者的患病信息；在信息检索的研究中，利用序列分类可将文件归类；通过蛋白质、DNA 序列的分类，有助于科学家探索蛋白质、DNA 的新功能[14]。

一般说来，序列可以看成是事件的集合，而每一个事件可以由符号或者能比较大小的实数表示。比如一段 DNA 序列可以表示成 $ACCCCCGT$, 一段简单的时间序列可以表示成 $\langle 0.2,0.3,0.5,0.9,\cdots\rangle$，两种序列的不同之处在于实数的大小是可以度量的，而一般而言符号之间的可度量性比较差，本文主要关注的是实数序列。对于符号序列，有一类的处理方法是借鉴实数序列，比如对于蛋白质序列和 DNA 序列，有一种基于序列对齐的距离度量方法[4]。

对于一般的序列分类任务来说，训练集里面的序列都含有描述其类属性的标签。对于语音识别任务来说，语料库里的语音其标签是其发出者；对于文档归类任务来说，文档的标签即是其所属类别。定义 L 为标签集，S 为待分类序列集，序列分类的任务就是通过对含有标签信息的训练集 T 的训练，得到一个可将 S 映射到 L 的分类器 C，对于进入 C 中的序列，都能够被贴上一个标签。可以表示成 $C : s \to l, s \in S, l \in L$。对于传统的分类任务来说，常用的分类器主要有 kNN，决策树、支持向量机[12]。

1.2 序列分类的主要方法

对于序列分类问题，因为其灵活性，很多的学者从不同的角度提出了大量的方法[1] [5] [10] [11] [13] [17]。本文主要介绍其中两种比较典型的方法，一种是基于序列距离度量的 Dynamic Time Warping(DTW) 方法[1] [2] [5] [7] [8] [10] [13]，另一种是基于特征选择的

shapelets 方法[17]。其中对 DTW 方法做重点介绍。

1.2.1 DTW 方法

对于分类任务而言，一种广泛使用而且特别简单的方法是 kNN，使用 kNN 时，关键是度量测试数据和训练数据的亲近与否，也就是需要定义一种距离。对于一般的实数序列，最常用的距离度量是欧氏距离，对于序列 s 和 s'，它们的欧式距离是：

$$dist\left(s, s^{'}\right) = \sqrt{\sum_{i=1}^{L}\left(s\left[i\right] - s'\left[i\right]\right)^{2}} \tag{1-1}$$

也就是相同索引的元素的差平方相加再开方。同样地，还可以定义其他诸如街区距离，最大值距离，这里需要重点强调的是，这些距离都是将同索引值元素对齐。

但是对于序列数据而言，这些距离有很大局限性，主要体现在以下两点：

1. 以上的距离度量都要求两序列长度相同，而这一点在实际问题中很难得到满足。比如在基因组分析中，很难保证基因序列的长度相同。

2. 设想这样一种情况。在步态分析中，同一测试者的步速可能不同，或者在某时间段上存在着加速和减速。那么对于其两段步态序列，比较相似的步态之间可能会有一定的时间差，而上面的这些距离测度只会将同一时刻的步态相比较。也就是说，上面的这些距离测度不能反映出序列比较中的错位。

上述的第一个缺点容易克服，两不同长度的时间序列，可以通过插值的方法使得序列等长，从而应用传统的距离度量方法。比如，对于两序列较短的那个，相邻点的之间可插入一定数目的点，插入点的数值可以用这两点的均值代替，以此小技巧，使得短序列扩展到与长序列长度。同样地，分段平均处理能够减小长序列至短序列长度[5]。但是对于对于大多数的序列而言，在描述序列之间的相似性时，更多的是考虑序列形态、走势方面的相似与否，而不会要求相似形态子序列的索引必须一致。传统的距离测度则只会比较同索比较。

DTW 方法能够很好地克服上述的两个缺点。

图1.1是 DTW 示意图，图中的两序列是某一长序列的两段，两子序列之间有一定的索引错位，示意图反映了有错位的两序列的对齐情况[3]。

图中的虚线表示了两序列点的对齐情况，这里对齐的两点指的是用来比较距离的两点，在传统的距离度量中，对齐的两点是索引值相同的两点。而从图中可以看出，对于 DTW 而言，点的对齐还参考了点邻近曲线的形态，而不仅仅局限于相同索引点，此外

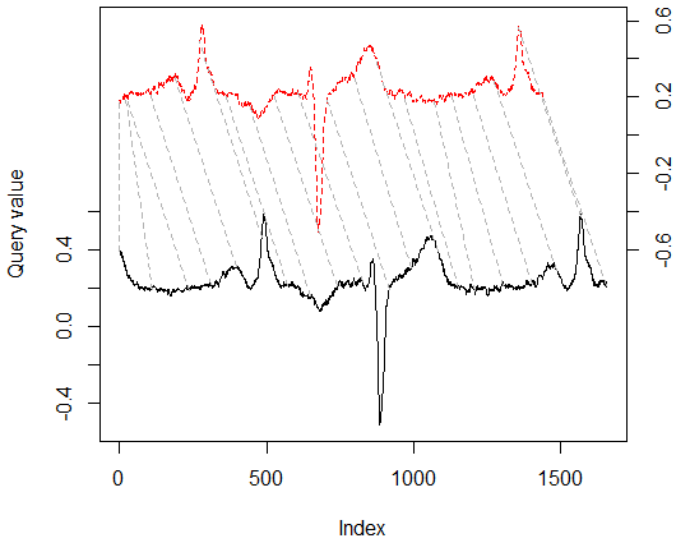


图 1.1: DTW 示意图

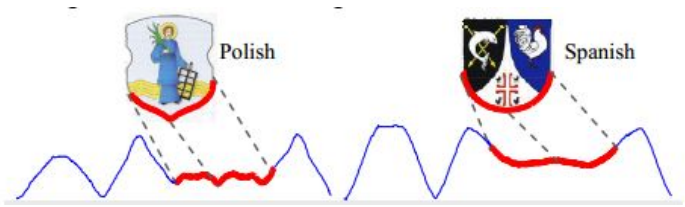


图 1.2: Shapelets 示意图

序列上的某点可以和另一个序列中的多个点对齐。因而 DTW 方法还能够直接处理两序列长度不同的情况。DTW 的具体求解将在下文中介绍。

1.2.2　Shapelets 方法

Shapelets 方法最早是由 (Ye *et al* 2009) 提出，作者经过大量实验，验证了该方法的优秀性[16]。

简单来说，选择序列中最能代表该序列的那一部分，比较其与 Shapelet 的相似度，就可对其分类。最具特征子序列如图1.2所示。

图中的样本是两个徽章，徽章的边缘可以转换成序列。对于这两个徽章而言，徽章的下部的轮廓最能够反映该徽章的特征，因而可以选择徽章下部轮廓对应的序列，比较其与 Shapelet 的相似度。可以看出，两徽章的下部明显不同，因而可以由下部序列之间的差异，利用分类器，对徽章进行分类。在这里，一般最常用的分类器是 kNN 分类器，它简单而有效，通常取 $k = 1$[13]。如图1.3所示。

Shapelets 方法有着广泛的应用，比如在植物学中，利用叶柄与叶片的角度，对植

图 1.3: 利用 Shapelets 进行 1NN 分类示意图

物进行分类；在考古学领域，通过对出土器件的特征部位的识别，利用决策树进行分类，能得到考古学发现[6]。如图1.4所示。

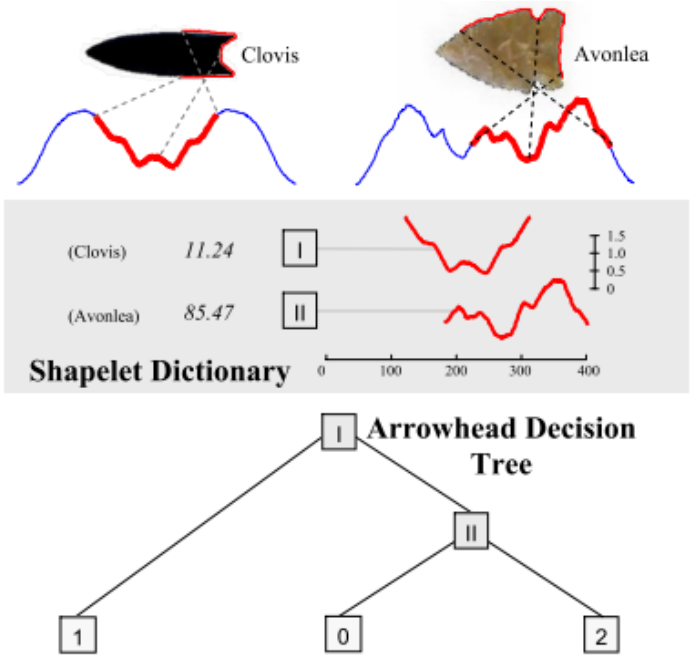


图 1.4: 基于 shapelets 的决策树方法对考古器件进行分类

第二章　DTW 方法的计算以及加速方法

2.1　DTW 方法的计算

2.1.1　不带有权重的 DTW 方法

首先定义两个序列 $Q = q_1, q_2, \cdots, q_i, \cdots, q_m$，$C = c_1, c_2, \cdots, c_j, \cdots, c_n$，并定义 Q 序列的第 i 个元素与 C 序列的第 j 个元素之间的距离为它们的欧式距离，即：

$$D\left(i^{th}, j^{th}\right) = d\left(q_i, c_j\right) = \left(q_i - c_j\right)^2 \tag{2-1}$$

DTW 方法中很重要的一点就是找到一条从两序列起点，到两序列终点的，能反映两序列对齐关系的 warping path。下面，首先给出 DTW 方法的 warping path 的示意图，如图2.1所示。示意图的左方和上方表示两个序列 Q 和 C，序列之间的方格是能够反映序

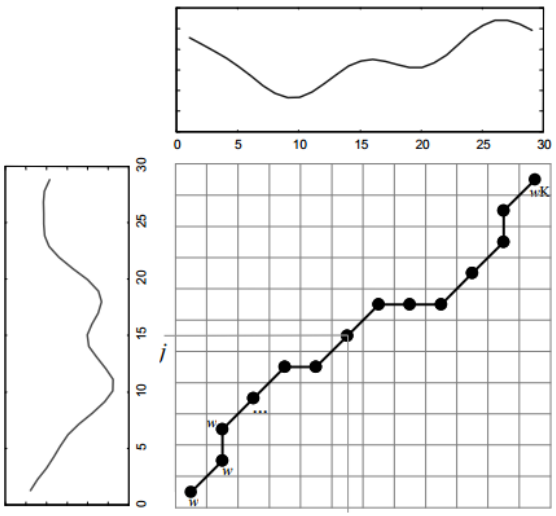


图 2.1: warping path 示意图

列对齐关系以及两序列对齐点之间距离信息的度量矩阵 M，M 的大小为 $m \times n$。M 中的黑色实线是 DTW 方法中的 warping path。warping path 记为

$$W = w_1, w_2, \cdots, w_k, \cdots, w_K \quad max(m, n) \leq K \leq m + n - 1 \tag{2-2}$$

w_k 表示路径上的第 k 个点，$w_k = (i,j)_k$，表示在路径的第 k 步，q_i 点和 c_j 点进行了对齐。反映在示意图上就是路径的第 k 步经过了 M 的坐标为 (i,j) 的方格。warping path 主要有以下三条约束：

1. **Boundary conditions:** $w_1 = (1,1)$ 以及 $w_K = (m,n)$。这条约束的含义是两序列最开始是起点对齐，最后是两序列的终点对齐。

2. **Continuity:** 给定路径的第 k 点 $w_k = (a,b)$ 以及路径的第 $k-1$ 点 $w_{k-1} = (a^{'},b^{'})$，那么 $a - a^{'} \leq 1$ 以及 $b - b^{'} \leq 1$。也就是说，路径游走的下一步点只能是当前点周围的那个点而不能跳跃。

3. **Monotonicity:** 给定路径的第 k 点 $w_k = (a,b)$ 以及路径的第 $k-1$ 点 $w_{k-1} = (a^{'},b^{'})$，那么 $a - a^{'} \geq 0$ 以及 $b - b^{'} \geq 0$。表示路径只能向前游走。

若 $w_k = (i,j)$，那么根据上面的三条约束，路径下一步游走的点只能是 $w_{k+1} = (i+1,j+1)$ 或 $w_{k+1} = (i+1,j)$ 或 $w_{k+1} = (i,j+1)$。在这三个约束下，DTW 方法需要的优化目标是：

$$DTW(W) = \min_{W} \left\{ \sum_{k=1}^{K} d\left(w_{ki}, w_{kj}\right) \right\} \tag{2-3}$$

(w_{ki}, w_{kj}) 表示路径 W 上的第 k 个点，说明第 k 步 q_i 和 c_j 点进行了对齐，$d\left(w_{ki}, w_{kj}\right)$ 表示 q_i 和 c_j 点的距离。也就是说，路径上的每点都对应着两序列对齐的点之间的距离，现在 DTW 的目标就是使路径上所有的这样的距离之和最短。求解这个最小化问题很简单，只需要用贪心的方法，使得每一次的迭代得到的距离和最小即可。定义这样的距离和为累计距离 $\gamma(i,j)$。直观意义是路径从距离矩阵的起始位置（左下方）蜿蜒到 (i,j) 方格的最小距离。路径蜿蜒到 (i,j) 方格的上一步是 $(i-1,j)$,$(i,j-1)$ 或 $(i-1,j-1)$，因而迭代公式为：

$$\gamma(i,j) = d(q_i, c_j) + \min\left(\gamma\left(i-1,j\right), \gamma\left(i,j-1\right), \gamma\left(i-1,j-1\right)\right) \tag{2-4}$$

根据这个迭代公式，DTW 方法的伪代码如 Algorithm1所示。

伪代码中 DTW 矩阵是累计距离矩阵，每个元素记录了 Q，C 两序列中对应元素之前的累计距离，矩阵的最后一个元素即为最终 Q 和 C 的 DTW 距离。

根据上面的迭代式，最终得到的仅仅是两序列最终的 DTW 距离，除此之外，还需要得到两序列的 DTW 对齐情况，也就是 DTW 路径，为了得到 DTW 路径，需要利用回溯的方法。如图2.2。回溯从累计距离矩阵的右上角开始，回溯的上一步是该方格左、

Algorithm 1 Calculate DTW

Require: $Q : array[1..m], C : array[1..n]$
Ensure: $DTW[m, n]$
1. $DTW := [0..m, 0..n]$
2. **for** $i := 0$ to m **do**
3. $\quad DTW[i, 0] := inf$
4. **end for**
5. **for** $j := 0$ to n **do**
6. $\quad DTW[0, j] := inf$
7. **end for**
8. $DTW[0, 0] := 0$
9.
10. **for** $i := 1$ to m **do**
11. $\quad$ **for** $j := 1$ to n **do**
12. $\quad\quad cost := d(Q[i], C[j])$
13. $\quad\quad DTW[i, j] := cost + min(DTW[i-1, j], DTW[i, j-1], DTW[i-1, j-1])$
14. $\quad$ **end for**
15. **end for**

下、斜下三个元素的最小值所在的方格，这样便能找到当前路径的上一步，通过不断这样地回溯，最终将到达路径的起始点，即得到 DTW 路径。

2.1.2　带有权重系数的 DTW 方法

对于 DTW 方法的路径走势，显然，路径总体斜向上游走，而不是呈阶梯形状游走更加合理。所以可以根据当前路径的走势，对路径的不同蜿蜒方向增加一个‘权重’，增加路径斜向上蜿蜒的可能性。此时，2-4中的最优化条件调整为:

$$DTW(W) = \min_{W} \left\{ \frac{\sum_{k=1}^{K} d\left(w_{ki}, w_{kj}\right) \cdot \alpha_k}{\sum_{k=1}^{K} \alpha_k} \right\} \tag{2-5}$$

$\sum_{k=1}^{K} \alpha_k$ 是正则化项，可以被设定为常数 C，因而2-5可以写成：

$$DTW(W) = \frac{1}{C} \min_{W} \left\{ \sum_{k=1}^{K} d\left(w_{ki}, w_{kj}\right) \cdot \alpha_k \right\} \tag{2-6}$$

α_k 表示第 k 步的权重系数，一般 α_k 有以下几种取值：

1. 对称形式：$\alpha_k = (i_k - i_{k-1}) + (j_k - j_{k-1})$，也就是说若路径下一步是沿着斜向上方向游走，那么权重系数 $\alpha_k = 2$，路径直向右或直向上游走，$\alpha = 1$，于是增加了路径斜向上游走的可能性。这种形式下，$C = m + n$。

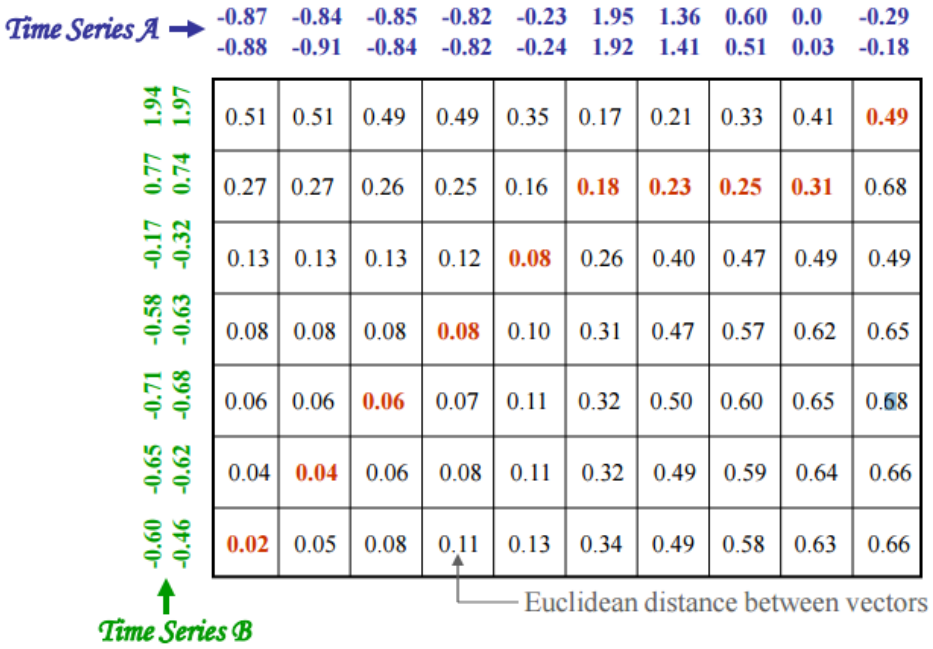


图 2.2: 回溯法得到 DTW 路径示意图

2. 非对称形式：

- $\alpha = (i_k - i_{k-1})$，这时 $C = m$。
- $\alpha = (j_k - j_{k-1})$，这时 $C = n$。

也就是说，在非对称情况下，路径朝上或朝右以及斜向游走，权重系数为 2。

对于带有权重系数的 DTW 方法，迭代公式只需要进行很小地调整。对于对称情形，迭代公式为：

$$\gamma(i,j) = d(q_i, c_j) + \min\left(\gamma\left(i-1,j\right), \gamma\left(i,j-1\right), \gamma\left(i-1,j-1\right) + d(q_i, c_j)\right) \tag{2-7}$$

两种非对称情形的迭代公式分别为：

$$\gamma(i,j) = d(q_i, c_j) + \min\left(\gamma\left(i-1,j\right) + d(q_i, c_j), \gamma\left(i,j-1\right), \gamma\left(i-1,j-1\right) + d(q_i, c_j)\right) \tag{2-8}$$

以及

$$\gamma(i,j) = d(q_i, c_j) + \min\left(\gamma\left(i-1,j\right), \gamma\left(i,j-1\right) + d(q_i, c_j), \gamma\left(i-1,j-1\right) + d(q_i, c_j)\right) \tag{2-9}$$

在实际的 DTW 方法的应用中，可以根据实际情况，对路径的每一步赋予特定的权

重，而这只需要在迭代公式上做很小的修改。

2.2　DTW 方法的加速技巧

从上文中可以看到 DTW 方法的时间和空间复杂度都为 $O(n^2)$，该节主要关注 DTW 方法的加速技巧。首先提出两种小技巧，最后引入能够将 DTW 时间和空间复杂度降低一个数量级的方法——FastDTW[10]。

2.2.1　warp window

虽然 DTW 方法能够对两序列中不同索引的元素进行对齐，但是直观上看，将两索引值相差过大的元素对齐在一起显然不合理。反映在距离矩阵上，也就是，路径不能偏离距离矩阵的对角线过大。基于此，可以定义 warp window，使得路径蜿蜒在 warp window 所限定的区域之中。warp window 如图2.3 所示。不使用 warp window，需要计

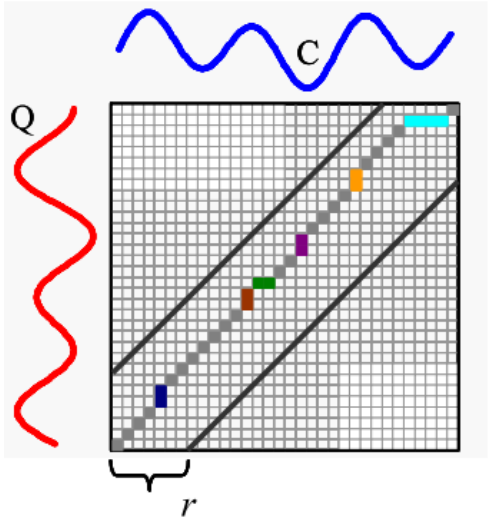


图 2.3: warp window 示意图

算整个整个距离矩阵，也就是说，需要填充整个距离矩阵。利用 warp window，只需要填充 warp window 所限定的、对角线周围的方格即可。所以引入 warp window，在一定的程度上，降低了计算量。此外，因为引入 warp window 后，路径的蜿蜒更符合直观实际，因而理论上说，引入 warp window 能够提高 DTW 方法的精确度。引入 warp window 后的 DTW 方法的伪代码如下所示：

与之前不带有 warp window 的 DTW 方法的主要不同在第 11 行中，通过控制 j 的取值范围来控制路径在距离矩阵中的蜿蜒范围。

Algorithm 2 Calculate DTW with warp window

Require: $Q : array[1..m], C : array[1..n]$, w :warp window
Ensure: $DTW[m, n]$

```
DTW := array[0..m, 0..n]
w := max(w, |m − n|)
for i := 0 to m do
    for j := 0 to n do
        DTW[i, j] := inf
    end for
end for
DTW[0, 0] := 0

for i := 1 to n do
    for j := max(1, i − w) to min(m, i + w) do
        cost := d(Q[i], C[j])
        DTW[i, j] := cost + min(DTW[i − 1, j], DTW[i, j − 1], DTW[i − 1, j − 1])
    end for
end for
```

2.2.2 序列的分段平均表示法

本节中的加快 DTW 计算的方法是从减小序列长度的角度出发的，序列的分段平均表示法[5] 是一种常用的有效减小序列长度的方法。首先介绍序列的分段平均表示法。给定序列 $X = x_1 x_2 \cdots x_n$，那么它的分段平均表示下的序列 $\bar{X} = \bar{x_1}\bar{x_2} \cdots \bar{x_N}$，这里假定 n 能被 N 整除。其中 $\bar{X}$ 的第 i 个元素是按照下面的计算式得到的：

$$\overline{x}_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} x_j \tag{2-10}$$

(a) 原始序列　　(b) 分段平均处理后的新序列

图 2.4: 序列分段平均处理示意图

也就是说 $\bar{X}$ 序列中的元素是对 X 序列中对应范围的元素求均值得到。经过序列分段平均处理后，再对新的较短的序列应用 DTW 方法，如图2.4所示。

在某些情况下，原始序列中的某些元素是序列中的噪声，经过序列分段平均处理，能从某种程度上消除序列噪声元素的影响，提取出序列特性。从这个角度看，对于某些序列，应用较为合适的序列分段平均处理，能够很大地减小序列长度，从而极大的加快 DTW 的计算，并且可以增加 DTW 方法的精度。

2.2.3　FastDTW 方法

FastDTW 方法简介

上面介绍的两种方法只能从 DTW 的实际应用经验上加快 DTW 方法的计算，本节引入一种新的能够从算法的角度有效地把 DTW 方法的时间以及空间复杂度从 $O(n^2)$ 降低到 $O(n)$ 的方法——FastDTW[10]。FastDTW 的思想是首先降低距离矩阵的分辨率，在较低分辨率下，得到 DTW 路径，然后对较低分辨率的路径所经过的方格，得到在较高分辨率下路径所能够走的方格的限制。算法主要包括以下三个关键步骤：

1. **粗化**：首先利用上面所提到的序列分段平均表示法，得到较短的序列。假设较短序列长度为原序列长度的 $\frac{N}{n}$，那么这些短序列构成的距离矩阵的宽度为原距离矩阵宽度的 $\frac{N}{n}$，也就是说，现在得到 $\frac{1}{\frac{n}{N}}$ 分辨率的距离矩阵。
2. **投影**：即找到低分辨率下的 DTW 路径。
3. **细化**：由低分辨率下 DTW 路径所经过的距离矩阵方格，找到较高分辨率下这些方格的对应方格，便找到高分辨率下 DTW 路径的约束。

FastDTW 方法的示意图如2.5所示。

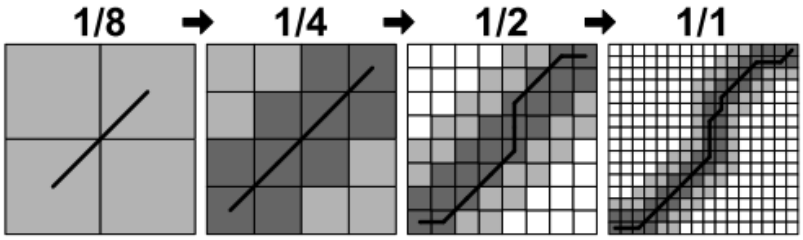


图 2.5: FastDTW 示意图

以图中的 $\frac{1}{2}$ 分辨率到 $\frac{1}{1}$ 分辨率为例。首先 $\frac{1}{2}$ 低分辨率是 $\frac{1}{1}$ 高分辨率下，通过对原始序列进行分段平均处理后得到的。在 $\frac{1}{2}$ 分辨率下可以得到如图所示的路径，路径周围黑色的方块是该分辨率下路径所走过的区域。在将分辨率调高到 $\frac{1}{1}$ 分辨率后，$\frac{1}{2}$ 分辨率下得到的黑色方块区域也相应地投影到 $\frac{1}{1}$ 分辨率中，即是 $\frac{1}{1}$ 分辨率中的黑色区域。但是在实际的操作中，为了使得得到的路径更为精确，这是会人为地设定一个参数：半径 r，它是一个倍率，表示将黑色区域扩大了多少倍。$\frac{1}{1}$ 分辨率下的黑色区域外围的灰色区域即是加入半径参数后对黑色区域的扩大，在该图所示的情况下，$r = 1$，在 $\frac{1}{1}$ 分辨率的距离矩阵中，路径蜿蜒区域被限定在黑色和灰色区域中。

在实际的 FastDTW 操作中，首先会把分辨率调低到一个合理的水平，然后逐步调高分辨率，每种分辨率下均能得到一条路径，最终将得到 $\frac{1}{1}$ 分辨率下的路径。FastDTW

对于每一种分辨率取最优路径，但是最终得到的不一定是和原始 DTW 方法相同的最优路径，因为从低分辨率路径得到的高分辨率情形下的路径限制方格，只是高分辨率中，路径最有可能经过的方格，真实的最优路径可能会‘突破’这些限制方格，所以 FastDTW 方法最终所得到的路径可能会和原始 DTW 路径有所不同。增大半径 r，会降低路径约束，使得 FastDTW 路径和原始 DTW 路径更为接近，但增加半径会使得需要填充的距离矩阵的方格更多，从而增加了计算量。

FastDTW 方法的时间、空间复杂度分析

首先计算 FastDTW 方法总共需要计算距离矩阵的元素，直观上看，也就是需要填充多少距离矩阵的方格。首先，考虑从 $\frac{1}{2}$ 分辨率到 $\frac{1}{1}$ 分辨率需要填充方格数目最多的情况，如图2.6所示：

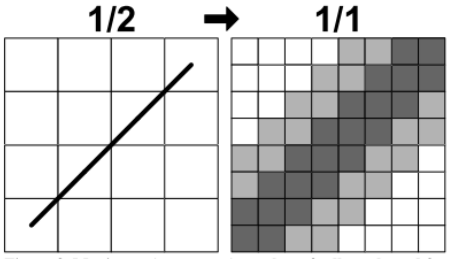


图 2.6: 最大填充方格示意图

$\frac{1}{1}$ 分辨率的图示中，黑色和灰色的区域都需要填充，因而在 $\frac{1}{1}$ 分辨率情形下，需要填充的最大方格数目上限为：$3N+2\cdot(2Nr)=N(4r+3)$。因为分辨率会有 $\frac{1}{1},\frac{1}{2},\frac{1}{2^2},\frac{1}{2^3},\cdots$ 这些，因而在所有分辨率下总共需要填充的方格数目为：

$$N\left(4r+3\right)+\frac{N}{2}N\left(4r+3\right)+\frac{N}{2^2}N\left(4r+3\right)+\frac{N}{2^3}N\left(4r+3\right)\cdots=2N\left(4r+3\right) \quad (2\text{-}11)$$

而时间复杂度由下面三部分组成：

1. 需要计算的方格数目。刚刚已经计算得到，是 $2N\left(4r+3\right)$。
2. 计算得到不同的分辨率。对于一段序列（长度为 N）来说，不同分辨率下需要的时间复杂度之和为：$N+\frac{N}{2}+\frac{N}{2^2}+\frac{N}{2^3}+\cdots=2N$，因为有两段序列，因而这部分的总时间复杂度为 $4N$。
3. 回溯得到不同分辨率下的路径。$\frac{1}{1}$ 分辨率下，路径长度最多为 $2N$，也就是说最多需要比较 $2N$ 次。所以所有分辨率下，回溯得到路径的时间复杂度是：$2N+\frac{2N}{2}+\frac{2N}{2^2}+\frac{2N}{2^3}+\cdots=4N$。

将这三项加起来，可得 FastDTW 方法的时间复杂度为：$N(8r+14)$。可以看到，FastDTW 是线性时间复杂度。

再来估计空间复杂度。空间复杂度由下面三部分组成：

1. 存储不同分辨率下的序列。因为不同分辨率下的序列长度分别为 $N, \frac{N}{2}, \frac{N}{2^2}, \frac{N}{2^3}, \cdots$，而有两条序列，因而存储不同分辨率下序列的空间复杂度为 $4N$。

2. 存储不同分辨率下的距离矩阵。前面已经估计到，总共需要填充的距离矩阵方格数目为 $N(4r+3)$。

3. 存储路径。前面也已经说明，存储路径的空间复杂度的上限为 $4N$。

因而 FastDTW 方法总的空间复杂度为:$N(4r+11)$。同样地，FastDTW 方法也为线性空间复杂度。当序列的长度比较小，也就是 N 比较小时，$N(8r+14)$ 以及 $N(4r+11)$ 的复杂度与 N^2 相比并没有优势，N 很大时，$N(8r+14)$ 以及 $N(4r+11)$ 将远远小于 N^2，因而，对于长序列，FastDTW 的时间、空间复杂度将远远优于原始 DTW 方法。

2.3　实验结果

2.3.1　实验 1

首先简单地使用人造数据比较欧几里得距离和 DTW 距离的分类精确度。实验有四类，每一类中训练集、测试集均都有四个序列，序列的长度均为 100，相邻类的两序列都有间隔参数 $interval$，序列是在相邻类间隔为 $interval$ 的常数序列中增加随机噪声而产生的，现通过 1NN 的方法对测试集中的序列分类，即将测试集中的序列归类到距离最近的那个序列所属的类，然后和测试集序列的 $label$ 相比较，计算得到分类精确度。最终的分类精确度是上面的数据集分类 100 次得到的平均精确度。

将间隔参数 $interval$ 从 0 变化到 2，可以得到相应的欧几里得距离和 DTW 距离的分类精确度曲线，如图2.7 所示。实验所得到的精确度曲线中，大部分情况下，DTW 距离的分类精度曲线都在欧几里得距离的分类精度曲线上面，这说明了 DTW 距离相对于传统欧几里得距离的优越性。

2.3.2　实验 2

实验 1 是使用人造数据进行实验，实验 2 把数据换为权威的官方数据。UCR Time Series Classification Archive 是加州大学河边分校相关研究人员收集的时间序列数据

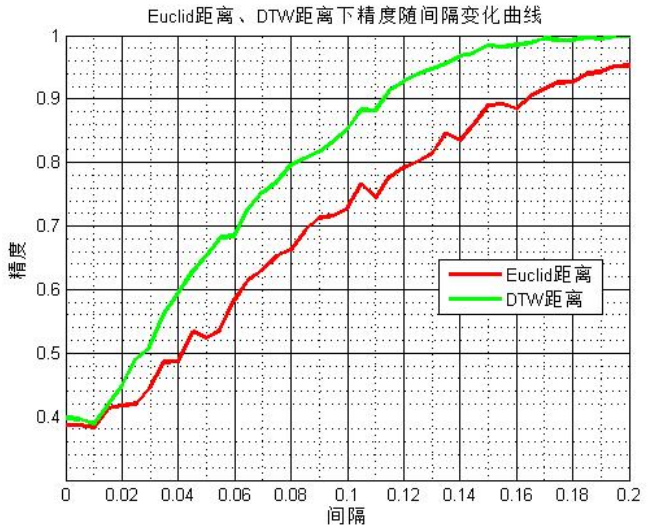


图 2.7: Euclid、DTW 距离下分类精确度随间隔变化曲线

集[15]，现采用其中的部分数据集，使用 DTW 以及欧几里得距离进行分类，得到结果如表所示。

表 2.1 UCR 数据集实验结果

name	Gun_Point	Plane	StrawBerry
lasses	2	7	2
training set size	50	105	370
test set size	150	105	613
sequence length	150	144	235
error rate (Euclid)	0.086667	0.038095	0.06199
error rate (DTW)	0.12	0	0.066884

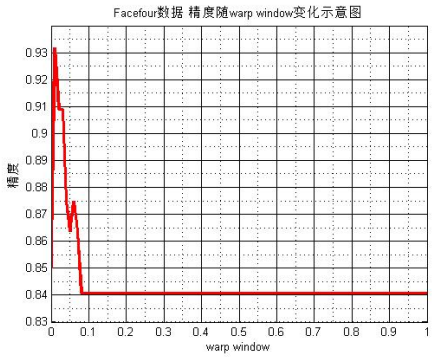
name	Computers	Trace	FaceFour	WordsSynonyms
lasses	2	4	4	25
training set size	250	100	24	267
test set size	250	100	88	638
sequence length	720	275	350	270
error rate (Euclid)	0.424	0.24	0.21591	0.38245
error rate (DTW)	0.332	0.01	0.15909	0.32445

从实验结果中的两个表格可以看出，在大多数情况下，DTW 方法的分类精确度都要高于传统的欧几里得距离。

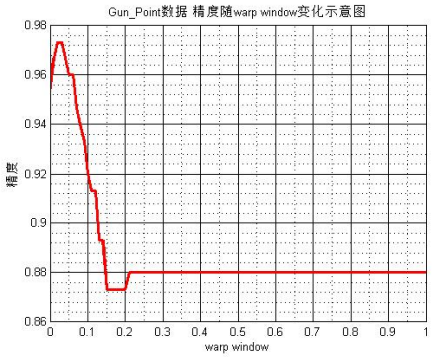
2.3.3 实验 3

上文已经说过，通过定义 warp window，不但能够减小计算量，而且能够在某种程度上增大分类精度。现在对于部分 UCR 数据集以及 $interval = 0.875$ 时的人造数据集，从 0 到 1 变化 warp window 的比例，得到各种比例下的分类精度，得到分类精度曲线，

如图2.8 所示。从分类精度随 warp window 变化曲线可以看出，较小的 warp window（大

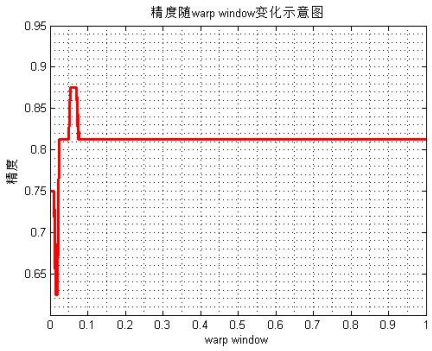


(a) facefour 数据



(b) gun_point 数据

(c) WordsSunonyms 数据



(d) interval 为 0.875 时的人造数据

图 2.8: 不同数据集分类精度随 warp window 变化曲线

部分情况下小于 10%）会提升分类精度。事实上，很多文献[15] [9] 也说明这点。

第三章 Shapelets 方法

上面介绍的 DTW 方法是从构造衡量两序列距离的角度分析，而本章通过序列最具特征子序列来完成分类。

3.1 Shapelets 简介以及定义

在日常的生活中，人们在对一个物体进行分类的时候，会根据这一类物体所具有的共同特征，比如我们想要区分一支笔是铅笔还是钢笔，我们只需根据笔尖的种类即可，笔尖带有圆珠，那么这支笔就是圆珠笔，笔尖中心有缝隙，那么这支笔就是钢笔。也就是说，对笔分类时，我们仅仅需要利用最能代表笔特征的部位——笔头即可。同样地，对于某些序列数据来说，我们也可以利用这种共同特征来对序列数据分类，这种体现共同特征的，序列集合中的某序列的子序列就是 Shapelets[17]。

一般而言，Shapelets 方法有以下三点好处：

- 得到可解释的结果：应用 Shapelets 方法的时候，结果中将包含一类序列的共同特征，这些共同特征将有助于对分类结果进行解释。

- 对于一些数据而言分类的结果更为精确。Shapelets 方法和大多数方法的不同之处在于，Shapelets 方法考虑的是序列的局部特征，而其他方法多考虑序列整体，这些方法比较容易受到序列中的无用部分以及序列中的噪声、失真等影响。例如把笔杆也考虑进分类中，那么分类结果必不如只考虑笔尖精确。

- Shapelets 方法的时间复杂度比较低。在得到 Shapelets 的情况下，分类的时间复杂度为 $O(ml)$，m 是待分类序列的长度，l 是 Shapelets 的长度。因为 Shapelets 是序列片段，因而一般情况下，l 都比较小。

一般可以借助信息增益来找到 Shapelets，给定顺序序列 $T = t_1, t_2, \cdots, t_m$，下面来逐步地给出 Shapelets 的定义。

Definition 1： 子序列. 序列 T 的长度为 l 的子序列 S 是由 T 中 l 个连续元素所组成的，$S = t_p, t_{p+1}, \cdots, t_{p+l-1} \quad p + l - 1 \leq m$。

Definition 2： 长度为 l 的子序列集合. 定义 $S_p^l = t_p, t_{p+1}, \cdots, t_{p+l-1}$ 是 T 的起始点为 t_p,长度为 l 的子序列。那么 T 的所有长度为 l 的子序列集合是 $\boldsymbol{S}_T^{|S|} = \left\{S_p^l, 1 \leq p \leq m - l + 1\right\}$。

Definition 3： 子序列 S 和长序列 T 之间的距离 $SubSeqDist(S, T)$. 定义 $SubSeqDist\,(T, S) = \min\left(Dist\left(S, S'\right), S' \in \boldsymbol{S}_T^{|S|}\right)$，即 S 和 T 之间的距离是 S 和 T 中所有等长子序列之间距离的最小值，如图3.1所示。

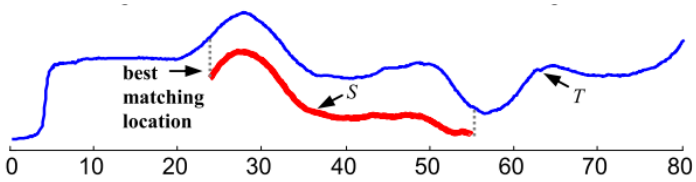


图 3.1: 子序列和长序列距离示意图

之所以把 S 和 T 之间的距离定义成 S 和 T 的子序列之间的距离，是因为 Shapelets 方法是对序列子序列比较的方法。

后文中将要看到，Shapelets 会将不同的类给分开，可以通过信息论中的信息增益来衡量这种分开的好坏。

Definition 4： 熵. 假定序列集合 $\boldsymbol{D}$ 由两类 A,B 组成，比例分别为 $p(A)$ 和 $p(B)$，那么 $\boldsymbol{D}$ 的熵的定义是 $I(\boldsymbol{D}) = -p(A)log(p(A)) - p(B)log(p(B))$。

有了熵的定义，那么就能得到 Shapelets 分类前后的信息增益。

Definition 5： 信息增益. 特定的划分策略 sp 将数据集 $\boldsymbol{D}$ 划分成两个子序列集 $\boldsymbol{D_1}$、$\boldsymbol{D_2}$，那么信息增益定义为：

$$Gain\,(sp) = I\,(\boldsymbol{D}) - \hat{I}\,(\boldsymbol{D}) \tag{3-1}$$

也就是：

$$Gain\,(sp) = I\,(D) - f\,(D_1)\,I\,(D_1) + f\,(D_2)\,I\,(D_2) \tag{3-2}$$

如图3.2所示，序列集 $\boldsymbol{D}$ 中有 10 个序列，包括两类，分别用红色和蓝色表示，每类的序列数目分别为 4 和 6，计算 Shapelets 候选序列和 $\boldsymbol{D}$ 中序列距离，按照大小关系排列在坐标轴上，如图所示的最优分割点的情况下，信息增益为：

$$\left[-\frac{6}{10}\log\left(\frac{6}{10}\right) - \frac{4}{10}\log\left(\frac{4}{10}\right)\right] - \left[\frac{5}{10}\left[-\frac{5}{5}\log\left(\frac{5}{5}\right)\right] + \frac{5}{10}\left[-\frac{4}{5}\log\left(\frac{4}{5}\right) - \frac{1}{5}\log\left(\frac{1}{5}\right)\right]\right] = 0.482 \tag{3-3}$$

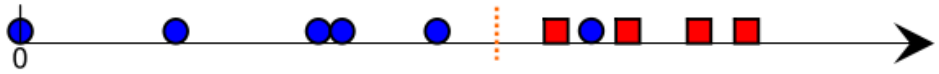


图 3.2: 信息增益计算的示意图

需要说明的是 Shapelet 是序列集 $\boldsymbol{D}$ 中所有子序列中的某个，那么 Shapelets 候选集就是 $\boldsymbol{D}$ 的所有子序列。

Definition 6： 最优分隔点 *(OSP).* 数据集 $\boldsymbol{D}$ 中包含两类序列 A 和 B。对于 Shapelets 候选序列 S，根据上面的距离的定义可以得到 S 与 $\boldsymbol{D}$ 中所有序列之间的距离，选择距离阈值 d_{th}，可以将 $\boldsymbol{D}$ 中的序列分为 $\boldsymbol{D}_1$ 和 $\boldsymbol{D}_2$ 两类，$\boldsymbol{D_1}$ 中的序列 $T_{1,i}$ 满足 $SubSeqDist(S, T_{1,i}) \leq d_{th}$，$\boldsymbol{D_2}$ 中的序列 $T_{2,i}$ 满足 $SubSeqDist(S, T_{2,i}) \geq d_{th}$。最优分隔点是所有分割点 d'_{th} 中，使得信息增益最大的那个分割点：

$$Gain\left(S, d_{OPS(D,S)}\right) \geq Gain\left(S, d'_{th}\right) \tag{3-4}$$

有了以上这些定义，最后将给出 Shapelets 的定义。

Definition 7： *Shapelets.* 给定含有两类序列的序列集合 $\boldsymbol{D}$，$Shapelet(\boldsymbol{D})$ 是 $\boldsymbol{D}$ 的子序列，满足对于 $\boldsymbol{D}$ 的所有子序列 S:

$$Gain\left(Shapelet\left(D\right), d_{OPS(D,Shapelet(D))}\right) \geq Gain\left(S, d_{OPS(D,S)}\right) \tag{3-5}$$

也就是说，在最优分割的情况下，利用 Shapelet 对序列集合进行分割，信息增益最大。

3.2　Shapelet 计算以及加速方法

3.2.1　Shapelet 的计算

利用 Shapelets 进行序列分类，首先需要找到序列集合的 Shapelets，算法如下。

伪代码的第一行是构造 Shapelets 候选集，该步中，Shapelets 候选集是长度介于 $MINLEN$，$MAXLEN$ 之间的所有子序列。然后遍历候选集中的所有候选序列，能够在最优分割情况下使得信息增益最大的候选序列即是序列集合的 Shapelet。

Algorithm 3 Brute force algorithm for finding shapelet

Require: dataset D,$MAXLEN$,$MINLEN$
Ensure: bsf_shapelet

```
candidates := GenerateCandidates(D,MAXLEN,MINLEN)
bsf_gain := 0
for S in candidates do
    gain:=CheckCandidate(D,S)
    if gain>bsf_gain then
        bsf_gain :=gain
        bsf_shapelet :=S
    end if
end for
```

3.2.2 Shapelet 计算的加速方法

在寻找 Shapelet 的过程中，需要遍历序列集所有的子序列，那么候选序列的总共数目将特别地大，而且在计算候选序列和原长序列的距离时，需要遍历原长序列中和候选序列等长的子序列，寻找最优分隔点也需要很大的计算量，所以寻找 Shapelet 的计算量特别大，下面介绍能够减小计算量的两种方法。

SubSeqDist 计算的提早舍弃

对于 Shapelets 候选序列 S 和原始序列 T，假设当前是取出 T 的子序列 S' 与 S 求距离。计算 S 与 S' 之间的距离，是按索引从小到大逐步计算的，我们可以把之间计算得到的最小距离给记录下来，当索引值推进到 S 和 S' 的某一步时，比较当前的距离和之前已记录的最小距离之间的大小，如果当前的距离比记录的最小距离大，那么就没有必要再拿 S' 和 S 比较下去，那么选择 T 的下一个子序列和 S 求距离。如图3.3 所示。

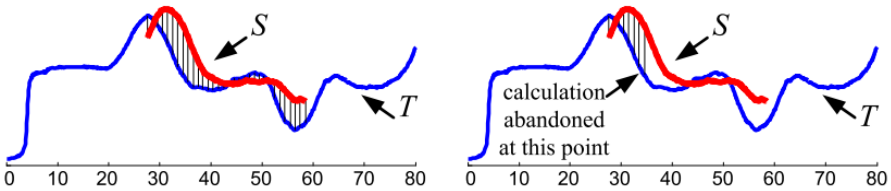


图 3.3: 计算候选序列和原始序列距离时的提早舍弃

计算熵的时候的提早舍弃

还是接着图3.2进行说明。对于另一个 Shapelets 候选序列 $\bar{S}$，假设当前已经计算了 $\bar{S}$ 和 $\boldsymbol{D}$ 中的 5 个序列（2 个红色序列，3 个蓝色序列）之间的距离，并按照大小顺序排列好，如图3.4所示。

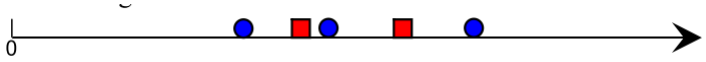


图 3.4: 候选序列与序列集中 5 个序列距离大小排列情况

对于下面的 5 个序列，当 3 个蓝色序列在最左边、2 个红色序列在最右边或者 3 个蓝色序列在最右边、2 个红色序列在最左边时，能获得的信息增益最大，如图3.5 所示，可以计算这种最大信息增益与之前记录的最大信息增益的大小关系，如果小于之前记录的最大信息增益，那么说明当前的 Shapelets 候选序列肯定不是最终的 Shapelet，所以应该选择 Shapelets 候选集中的下一个候选序列进行计算。

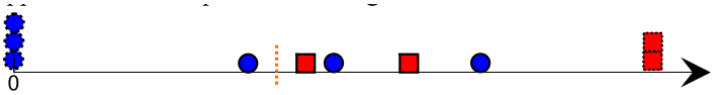


图 3.5: 使得熵最大的极端情况示意图

在得到序列集的 Shapelet 后，比较待分类序列和 Shapelet 之间的距离，根据距离远近，对待分类序列进行分类。

致谢

虽为致谢环境，其实就是一个 Chapter，为啥这么费事？因为，致谢一章没有编号。直接使用 `\chapter*{}` 的话，页眉又不符合工作手册要求，而且要往目录中添加该章节，还需要添加两行代码；为了简单快捷的设计出符合要求，又方便用户使用，只能借 `\backmatter` 模式和本模板自定义的 `\comtinuematter` 模式配合环境来做了。

参考文献

[1] Ghazi Al-Naymat, Sanjay Chawla, and Javid Taheri. SparseDTW: A novel approach to speed up dynamic time warping. *Conferences in Research and Practice in Information Technology Series*, 2009, 101(December 2003): 117–127.

[2] Gustavo E. a. P. a. Batista, Xiaoyue Wang, and Eamonn J Keogh. A Complexity-Invariant Distance Measure for Time Series. *SIAM International Conference on Data Mining*, 2011, pages 699–710.

[3] Toni Giorgino. Computing and Visualizing Dynamic Time Warping Alignments in R: The dtw Package. *Journal Of Statistical Software*, 2009, 31(7): 1–24.

[4] László Kaján, Attila Kertész-Farkas, Dino Franklin, Neli Ivanova, András Kocsor, and Sándor Pongor. Application of a simple likelihood ratio approximant to protein sequence classification. *Bioinformatics*, 2006, 22(23): 2865–2869.

[5] E. J Keogh and M. J Pazzani. Scaling up dynamic time warping for datamining applications. *Knowledge discovery and data mining*, 2000, In 6th ACM: 285–289.

[6] Eamonn Keogh. Ucr computational anthropology site. `http://www.cs.ucr.edu/~eamonn/anthropology/`.

[7] CHRISTINA LESLIE, ELEAZAR ESKIN, and WILLIAM STAFFORD NOBLE. the Spectrum Kernel: a String Kernel for Svm Protein Classification. *Biocomputing 2002*, 2001, pages 564–575.

[8] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing SAX: A novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 2007, 15(2): 107–144.

[9] Chotirat Ann Ratanamahatana and Eamonn Keogh. Three Myths about Dynamic Time Warping Data Mining. *Proceedings of the 2005 SIAM International Conference on Data Mining*, 2005, page 5.

[10] Stan Salvador and Philip Chan. FastDTW : Toward Accurate Dynamic Time Warping in Linear Time and Space. *Intelligent Data Analysis*, 2007, 11: 561–580.

[11] Sebastian Thrun, T O M Mitchell, and Kamal Nigam. Text Classification from Labeled and Unlabeled Documents using EM. 2000, 34: 103–134.

[12] Xindong Wu, Vipin Kumar, Quinlan J. Ross, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. *Top 10 algorithms in data mining.* volume 14. 2008.

[13] Xiaopeng Xi, Eamonn Keogh, Christian Shelton, Li Wei, and Chotirat Ann Ratanamahatana. Fast time series classification using numerosity reduction. *Proceedings of the 23rd international conference on Machine learning (ICML)*, 2006, pages 1033—-1040.

[14] Zhengzheng Xing, Jian Pei, and Eamonn Keogh. A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, 2010, nov, 12(1): 40.

[15] Bing Hu. Yanping Chen, Eamonn Keogh. The ucr time series classification archive. `http://www.cs.ucr.edu/~eamonn/time_series_data/`.

[16] L Ye. The time series shapelets webpage. `http://alumni.cs.ucr.edu/~lexiangy/shapelet.html`.

[17] Lexiang Ye and Eamonn Keogh. Time series shapelets: a new primitive for data mining. *Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining*, 2009, pages 947–956.