Subsequence Matching Based on Edit Distance Over Data Stream

LIANG Guang-min
The School of Electronics & Information Engineering，Shenzhen Polytechnic, Shenzhen 518055
gmliang@oa.szpt.net

Abstract: Data stream processing has recently attracted an increasing amount of interest. A variety of techniques currently exist for measuring the similarity between time series datasets. The researching of this paper is to monitor numerical stream, and to find subsequences that are similar to a given query sequence, under the ERP(Edit Distance With Real Penalty). In many applications such as network analysis and sensor monitoring, massive amounts of data arrive continuously and it is infeasible to save all the historical data. So we propose SM_ONSTREAM, pretreating the data first and then performing the subsequence matching. It can deal with noise and amplitude Shifting and time shifting of the time series. Experiments illustrate its high efficiency.

Key words: Subsequence matching; Edit Distance with Real Penalty; Data stream

基于编辑距离的数据流子序列匹配

梁广民
深圳职业技术学院电子与信息工程学院，深圳 518055
gmliang@oa.szpt.net

摘　要：本文研究的是在 ERP(Edit Distance With Real Penalty)距离下监测数字流，寻找与已知序列相似的子序列。许多应用如网络分析、传感器监测等中，数据源源不断地到达，由于受到资源的限制不可能存储所有的历史数据，并针对该类应用中噪音及序列的扭曲等因素的影响，提出了 SM_ONSTREAM 算法，先对数据进行预处理，并采用一种新颖的相似性度量指标，将算法复杂度降为 $O(N)$，完全满足数据流“一遍扫描”的实时性要求。最后，理论分析和实验结果表明，与现有的子序列匹配算法相比，本文提出的方法具有更强的性能和效率，更适合于数据流应用。

关键词：子序列匹配；实数惩罚的编辑距离；数据流

1 引言

数据流涉及许多重要的应用，如财政分析、网络监控，移动服务和传感器网络管理等，吸引了许多研究团体的兴趣，如理论的，数据库的，数据挖掘的以及网络的。在这些应用当中所面临的基础问题之一是有效地监测时间序列数据流。由于数据高速的到达以及大小不定和资源的有限性，不可避免地采用权衡的机制，实际上也不可能存储所有的历史数据，但必须保证对其进行高速的处理。在这样的应用当中就需要子序列的匹配机制去监测数据流。另外，在实际场合中，数据取样速度频繁改变和取样周期的变化，这种机制必须对噪音具有鲁棒性，并且能缩放时间轴。时间序列分析方法研究的关键问题之一是两序列相似性的度量模型的选择。

在这一领域，已有许多相似性度量模型，大概分为两大类。第一类是基于 L1 和 L2 范数的模型，例如欧式距离、动态时间扭曲（DTW）和实数惩罚的编辑距离（ERP）。

第二类是计算基于匹配阀值 ε 的相似性得分的模型，例如最长公共子串（LCSS）和实数序列的编辑距离（EDR）。以前的研究已表明第二类方法噪音，但它们是比较粗糙的度量，并且得出提出了 SPRING 算法进行数据流上的子序列匹配，此算法是在 DTW 下进

项目基金：深圳市科技计划项目，(QK 200608)
Fund: Shenzhen Municipal Science and Technology projects(QK 200608)

行的子序列匹配。DTW 对噪音比较敏感。而且在许多场合中，时间序列沿垂直时间轴上下移动即振幅平移，得到完全相似的序列，但按上述的距离函数计算都认为不相似，还有时间序列按比例缩放一定倍数，得到新的序列，我们一般认为它们是相似的，但按上述距离函数计算也认为是不相似的。针对以上问题，本文提出数据的预处理方案，采用 ERP 度量时间序列的相似性，进行数据流上的时间序列匹配。实验表明效果显著。

本文第 2 节介绍时间序列及相似性度量。第 3 节定义数据流上的子序列匹配问题，预处理方案以及算法的设计。第 4 节介绍仿真工作，并讨论实验结果。第 5 节给出结论并指出下一步的研究内容。

2 时间序列与相似性度量

2.1 时间序列

时间序列是由一记录值和记录时间组成的元素的有序集合，记为:

$X=(x_1=(v_1,t_1),x_2=(v_2,t_2),...,x_n=(v_n,t_n))$，元素 $x_i=(v_i,t_i)$ 表示时间序列在 t_i 时刻的记录值为 v_i，记录时刻为 t_i，是严格增加的 $(i<j \Leftrightarrow t_i<t_j)$。一般情况下，时间序列的采样间隔时间 $\Delta t=t_{i+1}-t_i$ 相等，可以看作 $t_1=0$，$\Delta t=1$，此时将时间序列 $X=(x_1=(v_1,t_1),x_2=(v_2,t_2),...,x_n=(v_n,t_n))$ 简记为 $X=(x_1,x_2,...,x_n)$。对于广义时间序列，记录值 v_i 可以是多种类型，包括离散符号、结构数据、多媒体数据等等。本文只考虑狭义时间序列，即 v_i 为实数值的情形。

2.2 相似性度量

2.2.1 几种相似性度量

时间序列的相似性度量研究是时间序列数据挖掘的基础问题。两条完全相同的时间序列几乎不存在，因此采用相似性度量来衡量时间序列之间的相似性。在许多研究领域，相似性度量常常与距离联系在一起.。距离表示两个对象之间的不相似的程度，大多数情况下，可以用距离度量来代替相似性度量。有些情况中，也用相似性得分来衡量时间序列的相似程度。欧氏距离是最常见的也是应用最广泛的一种距离。在计算两条时间序列之间的欧氏距离时，要求两条时间序列的长度相等。将长度为 n 的时间序列看作是 n 维欧式空间中的一个点，它的坐标值分别是时间序列在各个时刻的取值(按照时间先后顺序依次排列在坐标轴上)，那么两条长度为 n 的时间序列之间的欧氏距离就是 n 维空间中两个点之间的距离。其数学形式描述如表 1 中的（1），从公式上看，很明显它不能进行时间的伸缩与扭曲，而且对噪音很敏感。DTW(Dynamic time warping)定义如表 1 中的（2），它不需要两序列有相同的长度，通过重复利用以前的元素能进行局部的时间伸缩与扭曲。然而，欧氏距离和 DTW 都对噪音敏感，下面举例来说明这点。假定查询序列 $Q=(5,6,7,8)$，被查询序列 $R=(15,16,14,9)$、$S=(5,50,6,7,8)$、$P=(5,50,51,6,7)$。按相似性排序，正确的顺序为 S，P, R.，因为除了噪音外，S 和 Q 剩余的部分是完美的匹配。欧氏距离排序为 R、S、P, 序列在 DTW 下与欧氏距离排序相同。从序列的整体趋势来看，显然 S 比 P 更相似于 Q。这个例子说明对于噪音敏感的度量函数来说，相似的序列变得不相似了。长度分别为 m 和 n 的两序列的 LCSS 得分是通过表 1 中的(3)计算得来。LCSS 需要确定阀值 ε，用来判断序列中的两元素是否匹配，通过把两元素的距离量化成两个值 0 和 1 来处理噪音，从而避免了由噪音引起的大距离的影响，但没有考虑相似序列间空缺的大小。我们用同一个例子来说明这一点，假设 ，那么按 LCSS 排序，三个序列都与 Q 有相同的相似度。但是，序列 P 中的空缺要比 S 长，所以 S 较之 P 更相似于 Q。

2.2.2 ERP(Edit Distance with Real Penalty)

ERP 是建立在 ED（Edit Distance）的基础之上的，广泛应用于生物信息学，语音识别和运动物体轨迹的相似性研究。给定两个序列 R 和 S,它们的序列长度相同,都是 n。使用欧式距离来度量时间序列的相似性存在的问题是序列的长度必须相同，并且不支持局部时间轴的伸缩。为了处理时间轴的伸缩问题，我们可以借鉴字符串领域中的思想。一个字符串是一组元素组成的序列，其中的元素是字母表中的字母。通过最少的增、删和改操作数把两序列结队，在这三种操作当中，删操作可以被看成是在另一序列中增加一个字符，

978-1-935068-31-0 © 2010 SciRes.

Table 1. Comparison of some similarity measure
表 1. 几种相似性度量的比较

定 义	局部时间扭曲	噪音
$Eu(R,S)=\sqrt{\sum_{i=1}^{n}(r_i,s_i)^2}$ (1)		
$DTW(R,S)=\begin{cases}0 & if\ m=n=0\\ \infty & if\ m=0 or\ n=0\\ dist(r1,s1)+\min\{DTW(\mathrm{Re}st(R),\mathrm{Re}st(S)), & otherwise\\ DTW(\mathrm{Re}st(R),S),DTW(R,\mathrm{Re}st(S))\}\end{cases}$ (2)	√	
$LCSS(R,S)=\begin{cases}0 & if\ m=0 or\ n=0\\ LCSS(\mathrm{Re}st(R),\mathrm{Re}st(S))+1 & if\ \lvert r1-s1\rvert\le\varepsilon\\ \max\{LCSS(\mathrm{Re}st(R),S),LCSS(R,\mathrm{Re}st(S))\} & otherwise\end{cases}$ (3)	√	√
$EDR(A,B)=\begin{cases}n & if\ m=0\\ m & if\ n=0\\ \min\{EDR(\mathrm{Re}st(A),\mathrm{Re}st(B))+sub\cos t, & \\ EDR(\mathrm{Re}st(A),B)+1, & \\ EDR(A,\mathrm{Re}st(B))+1\} & otherwise\end{cases}$ (4)	√	√
$ERP(R,S)=\begin{cases}\sum_i^n\lvert s_i-g\rvert & if\ m=0\\ \sum_i^m\lvert r_i-g\rvert & if\ n=0\\ \min\{ERP(\mathrm{Re}st(R),\mathrm{Re}st(S))+dist_{erp}(r_1,s_1), & \\ ERP(\mathrm{Re}st(R),S)+dist_{erp}(r_1,gap), & otherwise\\ ERP(R,\mathrm{Re}st(S))+disterp(s_1,gap), & \end{cases}$ (5)	√	√

我们就把增加的这字符叫做空缺元素（gap element），这种距离就叫做串编辑距离（string edit distance）。引进空缺元素的代价或者称为距离设为 1。

$$dist(r_i,s_i)=\begin{cases}0\ if\ r_i=s_i\\ 1\ if\ r_i\ or\ s_i\ is\ a\ gap\\ 1\ otherwise\end{cases}\quad (1)$$

在公式（1）中我们在第二种情况中强调，如果在结队中引进一个空缺元素，那么其代价就是 1。把字符串推广为时间序列，序列中的元素就不再是符号，而是实数值。在许多应用当中，严格的相等没必要，例如，数对 $r_i=1,s_i=2$ 被认为更相似于数对 $r_i=1,s_i=1000$。考虑到实数值，那么有一种方法就是把相等松弛为在某一差值之内：

$$dist_{edr}(r_i,s_i)=\begin{cases}0 & if\ \lvert ri-si\rvert<\delta\\ 1 & if\ r_i\ or\ s_i\ is\ a\ gap\\ 1 & otherwise\end{cases}\quad (2)$$

公式（2）是公式（1）简单的推广，长度分别为 m 和 n 的两序列 R 和 S 的 EDR 距离定义如表 1 中的（4）。注意到公式（2），公式（4）中的最后一种情况可以简化为：

$$\min\begin{cases}EDR(\mathrm{Re}st(R),\mathrm{Re}st(S))+1,\\ EDR(\mathrm{Re}st(R),S)+1,\\ EDR(R,\mathrm{Re}st(S))+1\end{cases}$$

通过动态规划求得三种可能情况的最小和，而从本质上实施局部时间轴的伸缩。再对（1）推广，得出另外一种距离度量 DTW，

DTW 距离定义如下:

$$dist_{dtw}(r_i,s_i)=\begin{cases}\lvert ri-si\rvert & if\ r_i,s_i\ not\ gaps\\ \lvert ri-s_{i-1}\rvert & if\ s_i\ is\ a\ gap\\ \lvert s_i-r_{i-1}\rvert & if\ ri\ is\ a\ gap\end{cases}\quad (3)$$

DTW 不同于 EDR 主要表现在两个方面:

(1) DTW 没有使用 δ 阀值来松弛相等，实际上使用了 L1_norm。

(2) 在公式（3）没有明确的引入空缺的概念。在两序列结队的过程中，复制前一个元素来替代空缺的元素，因而空缺元素的代价就不是设为 1，而与前一个元素的 L1-norm。两序列的 DTW 距离就定义为表 1 中的（4）。

下面我们提出在 ERP 中，两非空缺元素就计算其 L1-norm 作为其惩罚，而使用一个常数来计算空缺元素间的距离。两元素间的 ERP 距离定义如下：

$$dist_{erp}(r_i,s_i)=\begin{cases}\lvert ri-si\rvert & if\ r_i,s_i\ not\ gaps\\ \lvert ri-g\rvert & if\ s_i\ is\ a\ gap\\ \lvert s_{i-1}-g\rvert & if\ ri\ is\ a\ gap\end{cases}\quad (4)$$

其中 g 的取值为 0。基于公式（4），两时间序列的 ERP 距离定义如表 1 中的(5)。从定义中可以看出，ERP 通过计算 L1-norm 距离来度量序列的相似性，要比 EDR 区分度大，并且引进 能进行时间轴上的伸缩。

3 数据流上的子序列匹配

3.1 问题的定义

离散的数据流 X 是半无限的数字序列 $x_1,x_2,\cdots,x_n,\cdots$，$x_n$ 是最新的值，n 随时间增大。若 $X[t_s,t_e]$ 表示从时刻 t_s 开始，t_e 结束的子序列，我们想寻找到与固定长度的查询序 Y 有高相似度的子序列。

当 X 是固定长度的序列，问题就可明确的表达为“最佳匹配”与“区域匹配”两种形式。固定长度的“最佳匹配”形式是一重要的里程碑。特别的，我们定义其子问题如下：

问题 1 最佳匹配查询（Best-match query）：

假定序列 X 和 Y 的长度分别为 n 和 m，寻找与 Y 有最小距离的子序列 $X[t_s,t_e]$ 就是在所有的可能序列 $X[t:j]$ 中寻找与 Y 距离最小的序列，即：

$$D(X[t_s:t_e],Y) \le D(X[t:j],Y)$$

$$(t=1,\cdots,n; j=t,\cdots,n)。$$

我们要解决的整个问题是数据序列X实际上是一半无限的子序列。在这种情形下，“最佳匹配查询”就没什么意义了，因为我们不能保证将会有比得到的序列更好的子序列。“区域匹配”就适合于数据流的情形。不过，值得注意的是，当查询序列 Y 与 X 的子序列匹配时，我们总期望有与此“局部最小”的最佳匹配重叠的几个其他匹配。因此，在标准的“区域匹配”中，我们提出附加第二个条件，目的是丢弃这些额外的匹配。这些匹配被怀疑有害：（a）他们将潜在地给用户以许多的冗余信息;（b）他们将降低算法的速度，迫使追踪和报道这些没用解。

问题 2 脱节查询（Disjoint query）：

给定数据流序列 X，在时刻 t 时的长度为 n,查询序列 Y 的长度为 m，阀值 ε，输出满足要求的所有子序列 $X[t_s,t_e]$：

（1） $D(X[t_s:t_e],Y) \le \varepsilon$

（2）在所有的重叠序列中，只输出局部最小序列，即，$D(X[t_s:t_e],Y)$ 是满足第一个条件的所有重叠序列中的距离最小者。

另外一个难题就是在时刻 t，找到能处理数据流 X 的一个新值并尽早的输出匹配序列的流形式解决方案。为了简化我们的阐述，我们将只注重在最佳匹配序列上，至于处理数据流的脱节查询,基本的思想都能用到这两种查询中。

3.2 数据的预处理

相似的时间序列由于其自身噪音与波动的特点，相似的时间序列会呈现多种变形，如振幅平移和伸缩、不连续、时间轴伸缩等等。振幅平移（Amplitude Shifting）是指两条形态相似的时间序列在不同的水平基准线上下波动。振幅伸缩（Amplitude Scaling）是指两条形态相似的时间序列以不同的振幅上下波动。不连续性（Discontinuity）是指两条时间序列除了在个别时间点或段出现间断外，其余大部分时间区间上形态相似。时间轴伸缩（Time Scaling）是指两条时间序列的波形相似，但是波形宽度按照同一个比例伸缩，也称为时间轴按比例伸缩。时间轴弯曲（Time Warping）是指两条时间序列的波形相似，但是波峰和波谷的位置并不是严格对齐，而是稍有偏差，是时间弯曲的一种特殊情况。为了消除振幅平移和振幅伸缩对时间序列相似性造成影响，（5）提出了规范化时间序列的预处理方法，规范化方法如下：设有时间序列 $X=(x_1,x_2,\cdots,x_n)$,集合 $\{x_1,x_2,\cdots,x_n\}$ 的均值和方差分别是 $\mu(X)$ 和 $\sigma(X)$,则规范化的时间序列为 $N(X)=(x_1^{'},x_2^{'},\cdots x_n^{'})$，其中 $x_i^{'}=(x_i-\mu(X))/\sigma(X)$，但是规范化以后的时间序列已经改变了原有的某些特征，在特定的应用如本文所研究的数据流上的子序列匹配上效果反而不佳。例如：设查询序列 $Y=(11,6,9.4)$，从数据流上截取到的序列为 $X=(5,12,6,10,6,5,1)$，两序列规范化以后得到序列

$$X^{'}=(-0.6963,2.7153,-0.2089,1.7406,$$
$$-0.2089-0.6963,-2.6458)$$

和 $Y^{'}=(2.0000,-0.8571,0.8571,-2.0)$。原序列的最佳匹配子序列为 $X_{sub}=(12,6,10,5)$。通过规范化以后原来匹配得很好的两个数据点如 $X(3)$ 和 $Y(2)$ 反倒匹配减弱了，而且如果把匹配阀值设为 $\varepsilon=0.5$，就变得不匹配了。原因就出在于规范化是对整个序列而言，所求的子序列以外的剩余部分影响了规范化后的数据特征，从而并不能达到预期的效果。针对上述问题，在本文中对时间序列的预处理时只对序列进行平移操作，即设 $X=(x_1,x_2,\cdots,x_n)$，处理后的序列为

978-1-935068-31-0 © 2010 SciRes.

$X^{'}=(x_1^{'},x_2^{'},\cdots x_n^{'})$,其中 $x^{'}=x_i-\mu(X)$ 。

3.3 算法的设计

3.3.1 算法检测过程

在这一节中，我们给出解决在 3.1 中描述的问题的算法。该算法能有效地检测到数据流上的高度相似的子序列，图 1 释了其检测过程。

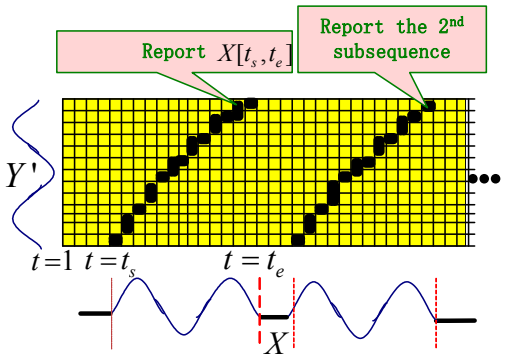

Figure 1. Algorithm testing process

图 1. 算法检测过程图

我们利用序列加前缀（“*”，star-padding）和子序列扭曲 $stwm(t,i)$ 矩阵(subsequence time warping matrix,STWM)的策略[2]，其中的序列前缀是一区间 $(-\infty,+\infty)$ ，在距离的计算中是不与考虑区间，也就是其ERP距离为零。假设 $Y=(y_1,y_2,\cdots,y_m)$ 是一查询序列，那么Y的加上前缀的序列为 Y' ：

$Y'=(y_0,y_1,y_2,\cdots,y_m)$ ， $y_0=(-\infty,+\infty)$

我们使用 Y' 来计算EDR距离，而不是在原来的序列 Y 上进行操作。序列加上前缀的两大好处：

（1）能判断出匹配序列的结束点

（2）告知与查询序列的ERP距离

子序列匹配矩阵中不仅存有子序列的匹配距离，而且还存储了子序列匹配的起始位置。假设两序列 $X=(x_1,x_2,\cdots,x_n)$ 和 $Y=(y_1,y_2,\cdots,y_m)$ ，我们就有其加上前缀的序列 $Y'=(y_0,y_1,y_2,\cdots,y_m)$ ，此 $y_0=(-\infty,+\infty)$ 。我们假设 E_i 和 $E_i^{'}$ 是一存储 m 个距离值的组， s_i 和 $s_i^{'}$ 是存储 m 个整数的数组，则序列间的 ERP 距离可递归的计算如下：

$$E_i=\min\begin{cases}E_{i-1}^{'}+dist(x_t,y_i)\\E_i^{'}+dist(y_i,g)\\E_{i-1}+dist(x_i,g)\end{cases}\qquad(6)$$

这里 $E_i=E(t,i)$ ， $E_i^{'}=E(t-1,i)$ （在时刻t），并且我们定义：

$$flag=\begin{cases}A & if\ \ Ei=E_{i-1}^{'}+dist(x_t,y_i)\\B & if\ \ Ei=E_i^{'}+dist(y_i,g)\\C & if\ \ Ei=E_{i-1}+dist(x_i,g)\end{cases}$$

相似我们可以获得序列的起始点：

$$s_i=\begin{cases}s_{i-1}^{'} & (flag=A)\\s_i^{'} & (flag=B)\\s_{i-1} & (flag=C)\end{cases}\qquad(7)$$

这里 $s_i=s(t,i)$ 和 $s_i^{'}=s(t-1,i)$ ，因而对每一时刻我们只要更新 m 个距离值和 m 个整数值。子序列匹配矩阵中，元素 $stwm(t,i)$ 中就同时存储了 E_i 和 s_i 。数据流上的最佳匹配问题的处理很直接的就可以利用(6)和(7)就能得出用户所需要的匹配序列。对于脱节查询最直接的算法就是:一经找到一个匹配序列，我们就报告出来并且初始化数组 E_i 。此算法满足问题 2 中第一个条件（即 $dist\le\varepsilon$ ），并且，如果用户想得到快速结果，此算法将非常有用。但，此算法不满足问题的第二个条件，事实上，如果存在距离都在 ε 内重叠序列时，此算法将丢失最优序列。因此，我们提出一个新的算法（SM_ONSTREAM），既能保证没有满足问题 2 的第 2 个条件的假的丢弃，而且还能较早的报告匹配序列。

3.3.2 算法分析

此算法通过确认当前最优序列不能被即将到来的子序列替换时才报告出子序列，此功能是通过考察重

叠序列的同时，跟踪最少距离 $E_{\min}$ 的变化。算法在数组 E_i 和 s_i 满足

$$\forall i, E_i > E_{\min} \vee s_i > p_{end}$$

的情况下报道出所得到的子序列，这意味着所得到的最优子序列不能被即将到达的序列替换，那么算法就在输出后给 $E_{\min}$ 和 s_i 重新赋值。算法只需一个单独的矩阵，在每一时刻 t 更新 $O(m)$ 个数字，因而时间复杂度和空间复杂度都为 $O(m)$。标准的 DTW 算法的时间复杂度和空间复杂度为 $O(m^2)$，因此我们的算法有明显的改进。

Algorithm SM_ONSTREAM

```
input: 时刻 t 时的新值 x_t
Output: 如果存在满足条件的序列就输出
对时间序列 (x_1, x_2, ... x_t) 进行预处理
for i=1 to m do
compute E_i and s_i by Equation (6) and (7);
if E_min ≤ ε then
    if ∀i, E_i > E_min ∨ s_i > p_end then
        Report( E_min, p_start, p_end );
        // Reset E_min and array of E_i
        E_min = ∞;
        for i=1 to m do
            if s_i ≤ p_end then
                E_i = ∞;
    endif
endif
if E_m ≤ ε ∧ E_m ≤ E_min then
    E_min = E_m; p_start = s_m; p_end = t;
endif
E_i' ← E_i; si ← s_i'
```

4 仿真实验

我们进行了 SM_ONSTREAM 算法的有效性和效率的实验。测试环境为 PentiumⅣ2.2,512M RAM, windows XP 和 JDK1.4.0。实验数据使用 eamonn 提供的 Earthquake 数据集 (http：//lib.stat.cmu.edu/general/tsa/.html)，该数据集包含了一条 4096 个点的序列。查询序列是从该序列中随机抽取的 100 个数据的子序列，并且在该序列的某处加以噪音或者对序列进行时间轴上的伸缩。下表是欧式距离、DTW 和 ERP 三种距离度量：实验中准确率与运行时间的比较结果（表 2）：

Table 2. Euclidean distance, DTW and ERP accuracy of three distance measures compared with the run-time results

表 2. 欧式距离、DTW 和 ERP 三种距离度量的准确率与运行时间结果比较

Distance	Mean accurac（%）	Run time（s）
Euclidean	13.89%	0.5
DTW	79.54%	13.7
ERP	86.12%	0.8

随机抽取 100 个点的序列，改变其个别点为噪音点，DTW 和 ERP 运行 100 次的结果：

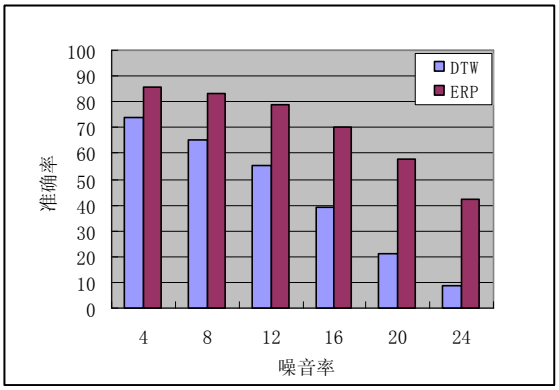

Figure 2. compare the accuracy of DTW and ERP rates at different noise

图2. DTW和ERP在不同噪音率下的准确率比较

5 总结与未来工作展望

本文详细地说明了时间序列以及相似性度量等相关概念，提出了基于 ERP 的数据流上的子序列匹配算法 SM_ONSTREAM.进一步将推广到多数据流的情形以及对噪音和时间伸缩更具鲁棒性的相似性

978-1-935068-31-0 © 2010 SciRes.

度量的研究。

References (参考文献)

[1] L. Chen, M. T. Özsu, and V. Oria. Robust and Fast Similarity Search for Moving Object Trajectories[J]. Bal-Timore, Maryland, USA, SIGMOD, 2005.

[2] Yasushi Sakurai, Christos Faloutsos, Masashi Yamam. Stream Monitoring under Time Warping Distance[C]. ICDE, 2007.

[3] Chen Wei man, Su-Liang, GAO Chun-ming, fast data stream sequence matching [J]. Computer Engineering and Applications, 2008 (36).
陈为满 ,苏亮, 高春鸣,数据流上快速子序列匹配[J]. 计算机工程与应用,2008(36)

[4] An Zhen zhou , Yang Jian, Wang Hong, Yu Ying, a new segment cutting DTW-based parallel algorithm [J], Computer Engineering and Applications 2007 (15)
安镇宙, 杨鉴, 王红, 余映. 一种新的基于并行分段裁剪的DTW 算法[J], 计算机工程与应用 2007(15).