データストリームのための時系列処理

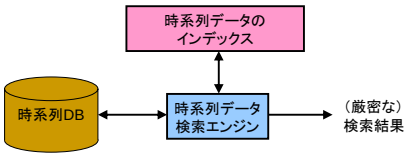
櫻井 保志
NTT サイバースペース研究所

講演内容

- 最近のデータストリーム処理の取り組み
  - タイムワーピング距離に基づくストリーム監視
    Y. Sakurai, C. Faloutsos, M. Yamamuro:
    Stream Monitoring under the Time Warping Distance.
    ICDE 2007
  - データストリームにおける遅延相関の検出
    Y. Sakurai, S. Papadimitriou and C. Faloutsos:
    BRAID: Stream Mining through Group Lag Correlations.
    SIGMOD Conference 2005: 599-610
- その他，最近の興味など

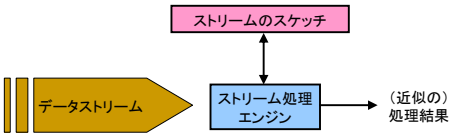
蓄積型とストリーム型

- 蓄積型の時系列データ処理
  - 例：類似シーケンスの検索処理
  - あらかじめ時系列DBからインデックスを作成
  - 問合せにより類似したシーケンスを探索
  - 検索結果は（一般的に）厳密



データストリーム処理

- データストリームにおける問合せ処理
  - データを受信する毎にシーケンスは長くなる（膨大なデータ）
  - ディスクのアクセスは避ける,しかしメモリ使用量には制限
  - ストリームの‘スケッチ’を作成，更新
  - 処理結果は（一般的に）近似



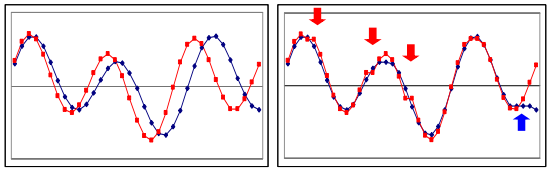
タイムワーピング距離に基づくストリーム監視

研究目的

- データストリームのアプリケーション
  - ネットワーク分析
  - センサー監視
  - 金融
  - ムービングオブジェクト
- 研究目的
  - データストリームをモニタリング
  - 固定長の問合せシーケンスと類似した部分シーケンスを検出
  - 距離基準：ダイナミックタイムワーピング (DTW)

ダイナミックタイムワーピング(DTW)

- 時間軸方向にシーケンス長を調整
  - シーケンス間の距離を最小化
  - シーケンスに 'stutter' を挿入
  - その後，(ユークリッド) 距離を計算



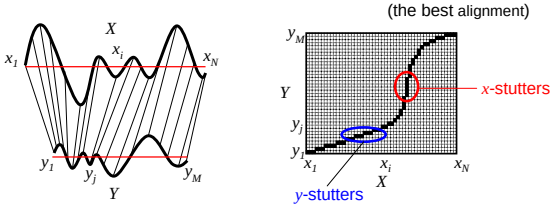
original　　'stutters'

7

ダイナミックタイムワーピング(DTW)

- DTWは動的計画法(dynamic programming)で計算
  - ワーピングパス：タイムワーピング行列の中のセルの集合

Optimum warping path
(the best alignment)



タイムワーピング行列

8

DTWの定義

- 二つのシーケンス X と Y が与えられたとき

$$X=(x_1, x_2, \ldots, x_n);\ Y=(y_1, y_2, \ldots, y_m)$$

DTWは動的計画法により計算

$$D(X,Y) = f(n,m)$$

$$f(t,i) = \|x_t - y_i\| + \min\begin{cases} f(t,i-1) & X\text{-stutter} \\ f(t-1,i) & Y\text{-stutter} \\ f(t-1,i-1) & \text{no stutter} \end{cases}$$

$$f(0,0)=0, \quad f(t,0)=f(0,i)=\infty$$

9

関連研究

- シーケンスのインデックス，部分シーケンスマッチング
  - Agrawal et al. (FODO 1998)
  - Keogh et al. (SIGMOD 2001)
  - Faloutsos et al. (SIGMOD 1994)
  - Moon et al. (SIGMOD 2002)
- DTWのためのシーケンスマッチング処理
  - Yi et al. (ICDE 1998)
  - Keogh (VLDB 2002)
  - Zhu et al. (SIGMOD 2003)
  - Sakurai et al. (PODS 2005)

10

関連研究

- データストリームのためのパターン検出
  - データストリームのクラスタリング
    Guha et al. (TKDE 2003)
  - ストリームのモニタリング
    Zhu et al. (VLDB 2002)
  - 情報予測
    Papadimitriou et al. (VLDB 2003)
  - 遅延相関
    Sakurai et al. (SIGMOD 2005)
- 従来研究の中で，ここで提起した問題を扱う研究はない

11

問題定義

- シーケンス $X=(x_1, x_2, \ldots, x_n)$
  - 固定長の問合せシーケンス Y から部分シーケンス$X[t_s,t_e]$ を検出
- 問題1：ベストマッチ問合せ (Best-match query)
  - とり得る全ての $X[t, i]$ の中で，Yからの距離が最小となる $X[t_s,t_e]$ を検出
  - 全ての $t=1,\ldots,n$ と $j=t,\ldots,n$ について

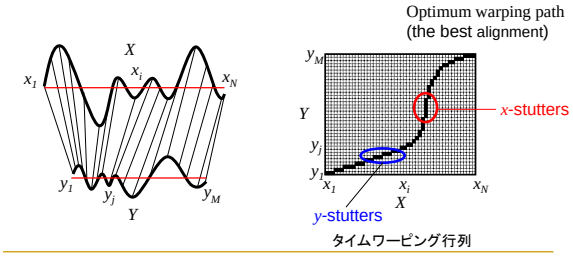
$$D(X[t_s:t_e],Y) \le D(X[t:j],Y)$$

  - ベストマッチ問合せはストリーム型よりも蓄積型
  - 範囲問合せの方がデータストリームの監視に適している

12

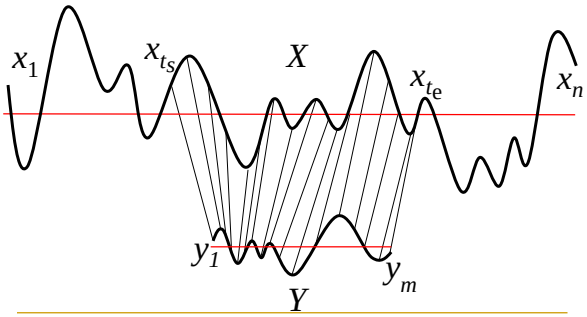
ダイナミックタイムワーピング(DTW)

- DTWは動的計画法(dynamic programming)で計算
  - ワーピングパス:タイムワーピング行列の中のセルの集合



13

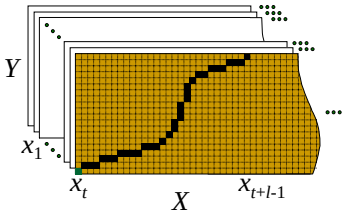
部分シーケンスマッチング



14

部分シーケンスマッチング

- データ受信毎にタイムワーピング行列を作成



15

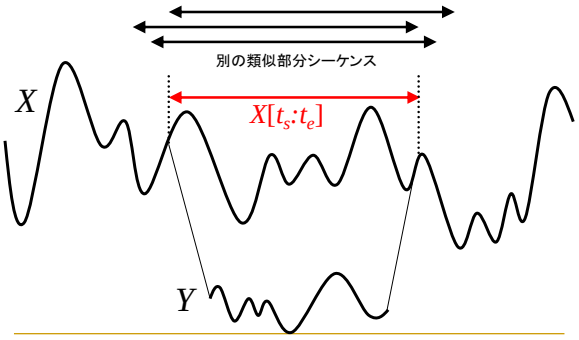
問題定義

- データストリームのための部分シーケンスマッチング
  - 範囲問合せ:距離が ε 以下の部分シーケンスを検出
  - 極小値をとる $X[t_s:t_e]$ と重複する部分シーケンスが数多く存在
- 2重の害
  - 利用者に冗長な情報を与えて悩ませる
  - 不必要な結果も報告させるために処理速度が低下
- 余分な部分シーケンスを除外

標準的な範囲問合せ ＋ 追加条件

16

部分シーケンスマッチング



17

問題定義

- 問題2:ディスジョイント問合せ (Disjoint query)
  - 閾値 ε として，二つの条件を満たす全ての $X[t_s:t_e]$ を検出
  - 条件1: $D(X[t_s:t_e],Y) \le \varepsilon$
  - 条件2: 極小値をとる部分シーケンス
    条件1を満たし，かつ重複する部分シーケンスの
    グループの中で $D(X[t_s:t_e],Y)$ が最小
  - ベストマッチ問合せと範囲問合せの組合せ
- データストリーム監視のために
  - '最適な'部分シーケンスを可能な限り早く出力

18

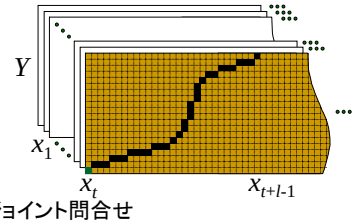
ナイーブな方法

- Super-Naïve
  - 全ての部分シーケンス $X[t_s:t_e]$ $(1 \le t_s \le t_e \le n)$ を考えて動的計画法を適用
  - 全ての計算量は $O(n^3m)$, 1レコード受信あたり $O(n^2m)$
  - $O(n^2)$ のタイムワーピング行列
- Naïve
  - 開始点が同じである部分シーケンスは行列を共有
  - 1レコード受信あたり $O(nm)$ の計算量
  - $O(n)$ のタイムワーピング行列

19

ナイーブな方法

- $O(n)$ 個のタイムワーピング行列



- ディスジョイント問合せ
  - 全ての部分シーケンスの距離を計算し，重複するものの中から最適な部分シーケンスを出力

20

提案手法SPRING

- アイデア1: スターパディング (Star-padding)
  - 1つの行列を用いて処理 (ナイーブな手法はm個)
  - 常に距離0を示す値 '∗' を Y に添える
  - $Y=(y_1, y_2, \ldots, y_m)$ の代わりにY'を用いて距離計算

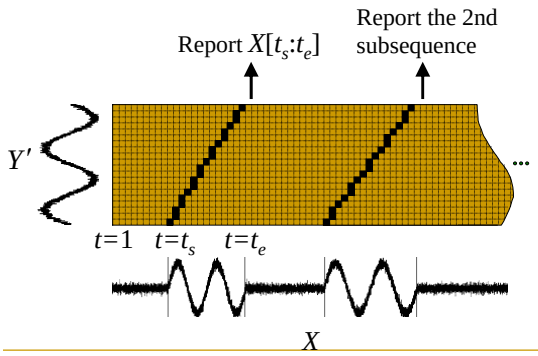
$$Y'=(y_0, y_1, y_2, \ldots, y_m)$$

$$y_0 = (-infinity, +infinity)$$

  - $O(m)$ の計算量とメモリ使用量（ナイーブは$O(nm)$ ）

21

SPRING



22

提案手法SPRING

- アイデア2: 拡張タイムワーピング行列
  - スターパディングの問題
  - 最適な部分シーケンスの位置情報を失ってしまう
  - スキャンの後，どの部分シーケンスが最小距離を出力したの？
- 拡張タイムワーピング行列
  - 部分シーケンスの距離値
  - その部分シーケンスの開始点
- スターパディングとの組合せによるストリーム処理
  - 不要な部分シーケンスを除去
  - 最小距離を示す部分シーケンスの位置を認識

23

アルゴリズム

- ディスジョイント問合せ
  - データ1つ受信する毎に行列のm個の要素（距離，開始点）を更新
  - ε 以下の部分シーケンスを検出すると，最適な部分シーケンスかどうかチェック
  - (a)(b)の条件を満たすときに現在の最適な部分シーケンスを出力

    (a) これから出現する候補部分シーケンスは最適でない

    (b) これから出現する候補部分シーケンスは現在の最適部分シーケンスと重複しない

24

SPRINGの例

- 上の値：距離，括弧内の値：開始点
- X=(5,12,6,10,6,5,13)　Y=(11,6,9,4)

$y_4 = 4$	54 (1)	110 (2)	14 (2)	38 (2)	6 (2)	7 (2)	88 (2)
$y_3 = 9$	53 (1)	46 (2)	10 (2)	2 (2)	10 (4)	17 (4)	18 (4)
$y_2 = 6$	37 (1)	37 (2)	1 (2)	17 (4)	1 (4)	2 (4)	51 (4)
$y_1 = 11$	36 (1)	1 (2)	25 (3)	1 (4)	25 (5)	36 (6)	4 (7)
x_t	5	12	6	10	6	5	13
t	1	2	3	4	5	6	7

25

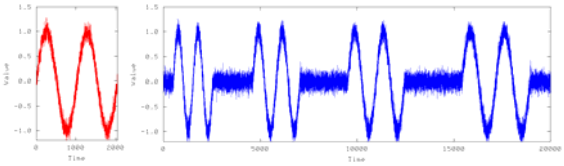
実験

- 実データと人工データを用いて実験
- 3つの評価項目
  - シーケンスのパターン検出はうまくいくのか？
  - シーケンス長 n が伸びると計算時間とメモリ量はどうなる？
  - ディスジョイント問合せについて，二つの提案手法に動作の違いはあるのか？

26

パターン検出

- 類似パターンをストリームから検出
  - *MaskedChirp*

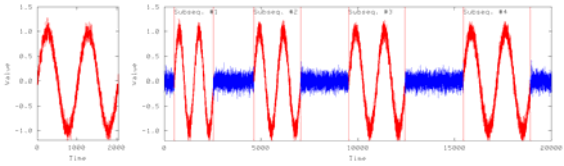


問合せシーケンス　　データストリーム

27

パターン検出

- 類似パターンをストリームから検出
  - *MaskedChirp*

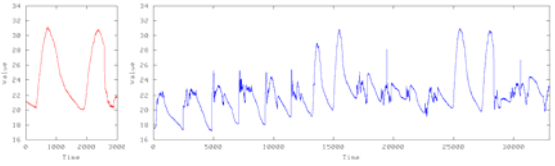


問合せシーケンス　　データストリーム

28

パターン検出

- 類似パターンをストリームから検出
  - *Temperature*

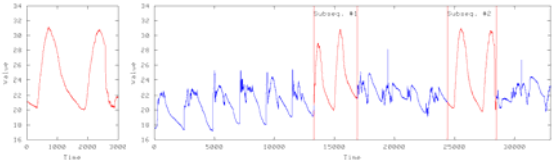


問合せシーケンス　　データストリーム

29

パターン検出

- 類似パターンをストリームから検出
  - *Temperature*

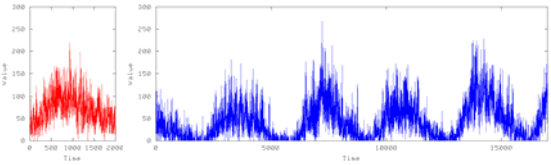


問合せシーケンス　　データストリーム

30

パターン検出

- 類似パターンをストリームから検出
  - *Sunspots*

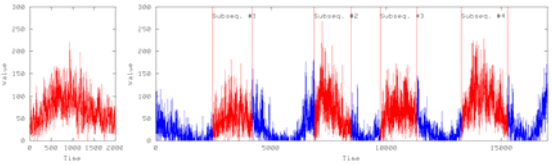


問合せシーケンス　　データストリーム

31

パターン検出

- 類似パターンをストリームから検出
  - *Sunspots*

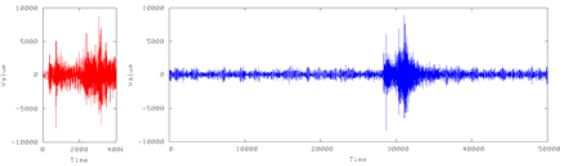


問合せシーケンス　　データストリーム

32

パターン検出

- 類似パターンをストリームから検出
  - *Kursk*

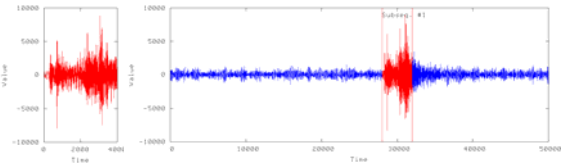


問合せシーケンス　　データストリーム

33

パターン検出

- 類似パターンをストリームから検出
  - *Kursk*

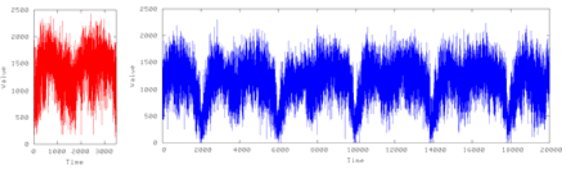


問合せシーケンス　　データストリーム

34

パターン検出

- 類似パターンをストリームから検出
  - *Automobile*

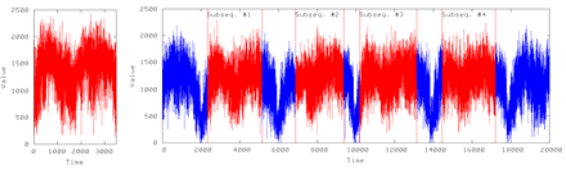


問合せシーケンス　　データストリーム

35

パターン検出

- 類似パターンをストリームから検出
  - *Automobile*

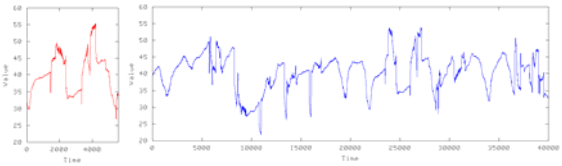


問合せシーケンス　　データストリーム

36

パターン検出

- 類似パターンをストリームから検出
  - *Humidity*

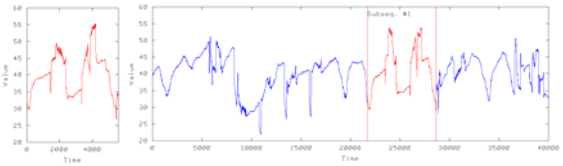


問合せシーケンス　　データストリーム

37

パターン検出

- 類似パターンをストリームから検出
  - *Humidity*

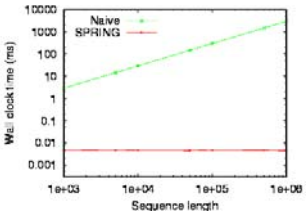


問合せシーケンス　　データストリーム

38

処理時間

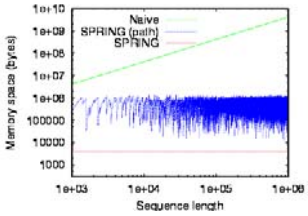
- 1レコードあたりの処理時間
  - ナイーブな手法は$O(nm)$
  - 提案手法は$O(m)$，シーケンス長nに依存しない



39

メモリ使用量

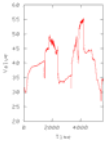
- タイムワーピング行列のためのメモリ使用量
  - ナイーブな手法は$O(nm)$
  - SPRINGは$O(m)$，シーケンス長nに依存しない
  - ワーピングパスに関する情報を出力する場合でも省メモリ化を達成



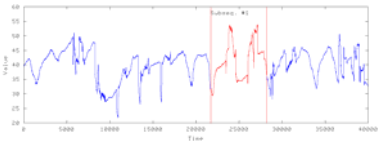
40

提案アルゴリズムの比較

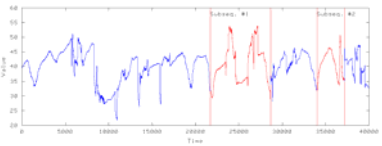
- 閾値εを変化
  - *SPRING-optimal*



問合せシーケンス



ε = 10,000

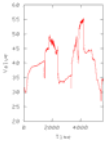


ε = 15,000

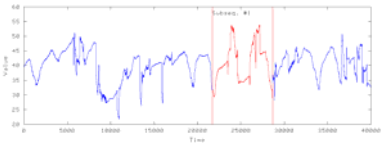
41

提案アルゴリズムの比較

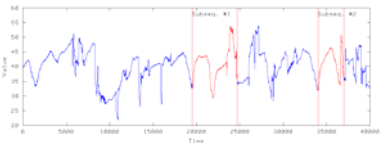
- 閾値εを変化
  - *SPRING-first*



問合せシーケンス



ε = 10,000



ε = 15,000

42

パターン認識とストリーム技術の融合

- データストリーム処理技術の映像への応用
- パターン検出の例
  - 人物にセンサーを取り付け（カーネギーメロン大学）
  - 運動量の類似性をもとにして動作を検出

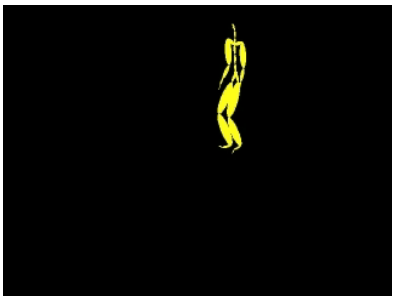
$$E_{rotation} = \frac{1}{2} I\omega^2$$

$E_{rotation}$: 角運動量，I: 慣性モーメント，ω: 角速度

- 娯楽のためのアプリケーション
  - リアルタイム映像作成

43

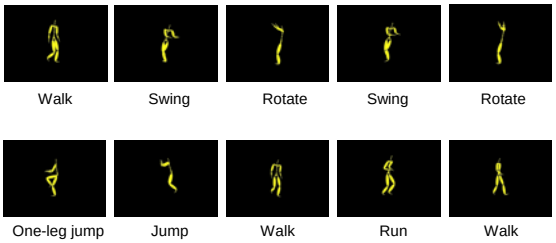
映像処理への応用



44

動作の高速な認識

- データストリームに含まれる動作



45

まとめ

- SPRING: DTWに基づくデータストリームのための部分シーケンスマッチング
- 高速，省メモリ
  - $O(m)$ の計算時間とメモリ空間，n に依存しない
- 精度
  - 検出漏れがないことを保証
- 蓄積型への適用も可能
  - 蓄積型の検索手法と組み合わせることが可能

46

データストリームにおける遅延相関の検出

研究目的

- データストリームのアプリケーション
  - ネットワーク分析
  - センサー監視
  - 金融
  - ムービングオブジェクト
- 研究目的
  - 複数のストリームをモニタリング
  - どのペアに遅延相関があるのかを検出
  - 遅延の値を報告（もし相関しているペアがあれば）

48

遅延相関

- 遅延相関の例
  - 金利が下がると数ヵ月後に住宅着工数が増加する

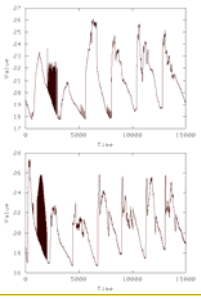
  - 水道水にフッ素を多く混入させると数年後に虫歯の患者数が減少する

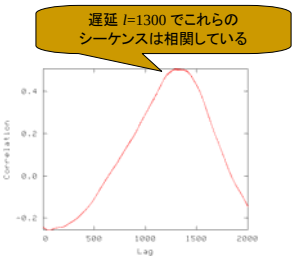
  - サーバー1のCPUの利用率が上昇すると数分後にサーバー2のCPU利用率も上昇する

49

遅延相関

- 遅延相関のあるシーケンス



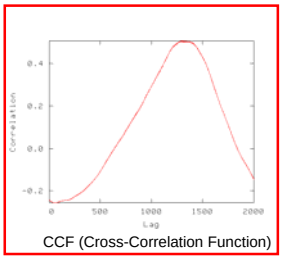


CCF (Cross-Correlation Function)

50

遅延相関

- 遅延相関のあるシーケンス
  - 高速
  - 省メモリ
  - 高精度
    （少ない誤差）

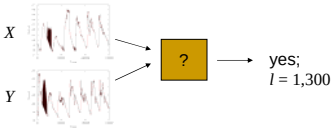


CCF (Cross-Correlation Function)

51

問題1：シーケンスのペア

- 伸長するシーケンス XとY について，
  - 遅延相関があるかどうか
  - もし遅延相関があれば，遅延の長さ l はどれくらいかを推定する

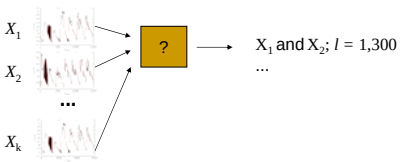


  - データストリーム（伸長するシーケンス）において，いつでも報告可能

52

問題2：k-way

- 与えられた k 個のシーケンス $X_1,\ldots,X_k$ について
  - どのペアに遅延相関があるのか
  - そのペア各々の遅延の値を推定する



  - いつでも，ストリーム上で．．．

53

提案手法，BRAID

- 特徴
  - いつでも処理可能，そして高速
    単位時間あたりの計算時間は一定
  - 省メモリ
    メモリ空間は，シーケンス長 n に対して $O(\log n)$
  - 高精度
    近似誤差は小さい

54

関連研究

- シーケンスのインデックス
  - Agrawal et al. (FODO 1993)
  - Faloutsos et al. (SIGMOD 1994)
  - Keogh et al. (SIGMOD 2001)
- データ圧縮（ウェーブレット変換，ランダム射影）
  - Gilbert et al. (VLDB 2001)
  - Guha et al. (VLDB 2004)
  - Dobra et al.(SIGMOD 2002)
  - Ganguly et al.(SIGMOD 2003)

55

関連研究

- データストリーム管理
  - Abadi et al. (VLDB Journal 2003)
  - Motwani et al. (CIDR 2003)
  - Chandrasekaran et al. (CIDR 2003)
  - Cranor et al. (SIGMOD 2003)

56

関連研究

- パターン検出
  - データストリームのクラスタリング
    Guha et al. (TKDE 2003)
  - ストリームのモニタリング
    Zhu et al. (VLDB 2002)
  - 情報予測
    Yi et al. (ICDE 2000)
    Papadimitriou et al. (VLDB 2003)
- 従来研究の中で，ここで提起した問題を扱う研究はない

57

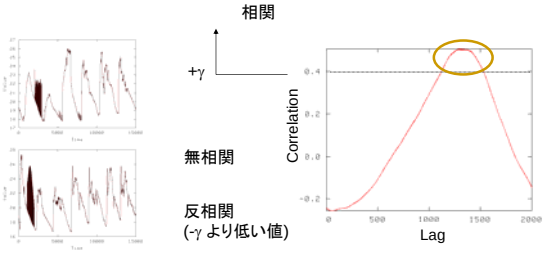
アウトライン

- 研究目的，関連研究
- 前提条件
- 提案手法
- 理論的な分析
- 実験結果

58

前提条件

- 遅延相関



CCF (Cross-Correlation Function)

59

前提条件

details

- ‘*score*’の定義，$R(l)$ の絶対値

$$score(l) = |R(l)|$$

$$R(l) = \frac{\sum_{t=l+1}^{n}(x_t - \bar{x})(y_{t-l} - \bar{y})}{\sqrt{\sum_{t=l+1}^{n}(x_t - \bar{x})^2}\sqrt{\sum_{t=1}^{n-l}(y_t - \bar{y})^2}}$$

- 遅延相関
  - しきい値 γ が与えられたとき，$score(l) > \gamma$
  - local maximum（極大値）
  - 遅延が最も少ない local maximum

60

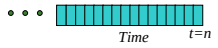
アウトライン

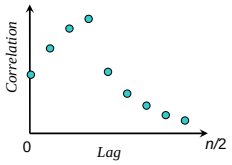
- 研究目的，関連研究
- 前提条件
- 提案手法
- 理論的な分析
- 実験結果

61

ナイーブな方法

- 高速化手法を使わない場合
  - 各々の遅延について相関係数を計算
    l = 0, 1, 2, 3, …, n/2
- しかし
  - メモリ空間 $O(n)$
  - 計算時間 $O(n^2)$
    もしくは $O(n \log n)$ w/ FFT





62

アイデア（1）

- インクリメンタルな計算
  - “the correlation coefficient of two sequences is ‘algebraic’” => 相関値はインクリメンタルに計算可能
- 6つの基本統計情報，‘sufficient statistics’
  - シーケンス長 n
  - シーケンス X の和，X の2乗和
  - シーケンス Y の和，Y の2乗和
  - X とシフトしたY の内積

63

Main Idea (1)

details

- Incremental computing:
  - Sequence length n
  - Sum of X : $Sx(1,n) = \sum_{t=1}^{n} x_t$
  - Square sum of X : $Sxx(1,n) = \sum_{t=1}^{n} x_t^2$
  - Inner-product for X and the shifted Y : $Sxy(l) = \sum_{t=l+1}^{n} x_t y_{t-l}$
  - Compute $R(l)$ incrementally:

$$R(l) = \frac{C(l)}{\sqrt{Vx(l+1,n) \cdot Vy(1,n-l)}}$$

  - Covariance of X and Y:

$$C(l) = Sxy(l) - \frac{Sx(l+1,n) \cdot Sy(1,n-l)}{n-l}$$

  - Variance of X:

$$Vx(l+1,n) = Sxx(l+1,n) - \frac{(Sx(l+1,n))^2}{n-l}$$

64

アイデア（1）

- 計算量

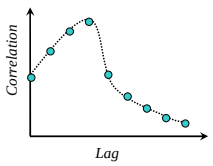
	Naive	Naive (incremental)	BRAID
空間計算量	$O(n)$	$O(n)$	
時間計算量	$O(n \log n)$	$O(n)$	

もっと良くならない？

65

アイデア（2）

- 等比数列に基づくCCFのサンプリング

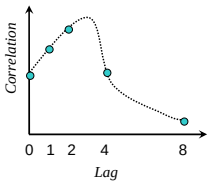


66

アイデア（2）

- 等比数列に基づくCCFのサンプリング
- 遅延 $l = 0, 1, 2, 4, \ldots 2^h$ について，相関係数を計算

$O(\log n)$ estimations



67

アイデア（2）

- 等比数列に基づくCCFのサンプリング

	Naive	Naive (incremental)	BRAID
空間計算量	$O(n)$	$O(n)$	
時間計算量	$O(n \log n)$	$O(n)$	$O(\log n)$

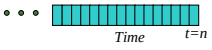
- 例えば遅延の最大値を $n/2$ とすると，$O(n)$ のメモリが必要

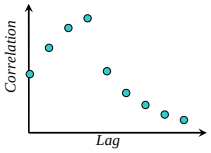
68

アイデア（3）

- シーケンスの平滑化

ナイーブな方法

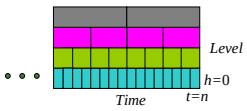


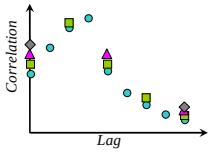


69

アイデア（3）

- シーケンスの平滑化
  - 各レベルの窓の平均
  - 基本統計情報は平均値から計算
  - 遅延相関は平均値の基本統計情報から計算
  - しかし，冗長な情報がある …

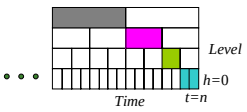


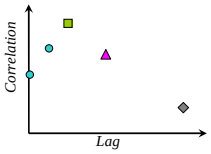


70

アイデア（2）と（3）

- サンプリング + 平滑化
  - カラーで示した窓のみを使う
  - 等比数列の遅延のみ計算
    $l=\{0,1,2,4,8,\ldots,2^h,\ldots\}$

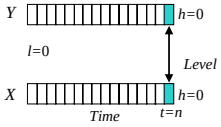


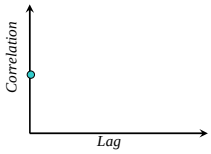


71

アイデア（2）と（3）

- サンプリング + 平滑化
  - カラーで示した窓のみを使う
  - 等比数列の遅延のみ計算
    $l=\{0,1,2,4,8,\ldots,2^h,\ldots\}$

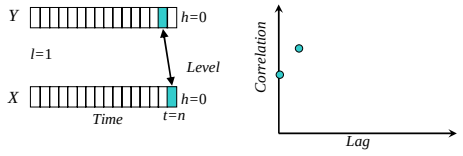




72

アイデア(2)と(3)

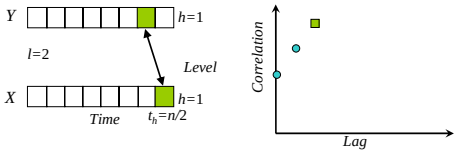
- サンプリング + 平滑化
  - カラーで示した窓のみを使う
  - 等比数列の遅延のみ計算
    l={0,1,2,4,8,…,2^h,…}



73

アイデア(2)と(3)

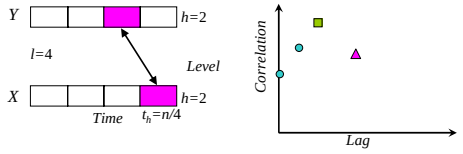
- サンプリング + 平滑化
  - カラーで示した窓のみを使う
  - 等比数列の遅延のみ計算
    l={0,1,2,4,8,…,2^h,…}



74

アイデア(2)と(3)

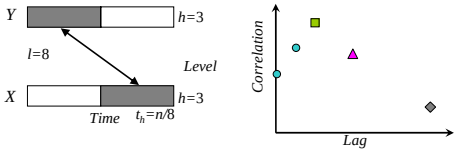
- サンプリング + 平滑化
  - カラーで示した窓のみを使う
  - 等比数列の遅延のみ計算
    l={0,1,2,4,8,…,2^h,…}



75

アイデア(2)と(3)

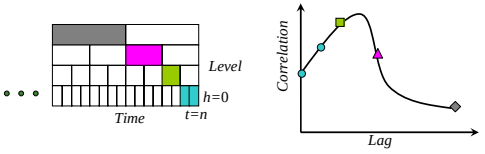
- サンプリング + 平滑化
  - カラーで示した窓のみを使う
  - 等比数列の遅延のみ計算
    l={0,1,2,4,8,…,2^h,…}



76

アイデア(2)と(3)

- サンプリング + 平滑化
  - カラーで示した窓のみを使う
  - 等比数列の遅延のみ計算
    l={0,1,2,4,8,…,2^h,…}
  - 3次元スプラインを用いて相関値を補間



77

結果として…

- 計算量

	Naive	Naive (incremental)	BRAID
空間計算量	$O(n)$	$O(n)$	$O(\log n)$
時間計算量	$O(n \log n)$	$O(n)$	$O(1)$ *

(*) 時間計算量 $O(\log n)$
ならし計算量 $O(1)$

78

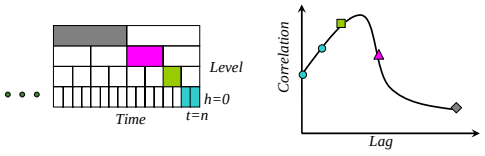
アウトライン

- 研究目的，関連研究
- 前提条件
- 提案手法
  - 提案手法の発展
- 理論的な分析
- 実験結果

79

精度の向上

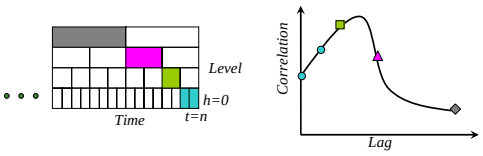
- Q: どうすれば 2^h よりも細かくサンプリングできる？



80

精度の向上

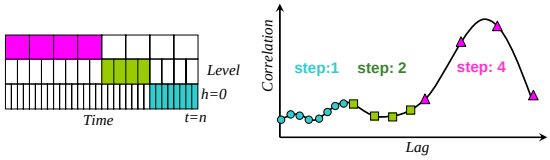
- Q: どうすれば 2^h よりも細かくサンプリングできる？
- A: 等比数列と等差数列の併用



81

Enhanced BRAID

- 基本スキームは b=1（各レベルに一つ）
- 改良スキームは b>1
  - b=4 の例
  - 等比数列と等差数列を併用したCCFのサンプリング
    l={0,1,…,7;8,10,12,14;16,20,24,28;32,40,…}



82

アウトライン

details

- 研究目的，関連研究
- 前提条件
- 提案手法
- 理論的な分析
- 実験結果

83

理論的な分析 – 精度

details

- 平滑化の効果

  低周波で構成されたシーケンスに対しては，
  平滑化による誤差は小さい

- サンプリングの効果

  標本化定理（ナイキスト）を満たすとき，
  BRAIDは誤差を生まない

84

理論的な分析 – 精度

details

- サンプリングの効果
  - 概要：標本化定理（ナイキスト）を満たすとき，BRAIDは誤差を生まない
  - 詳細：

以下を満たすとき，BRAIDは必ず遅延相関を見つけることができる

$$0 \le l \le \frac{2b}{f_R}$$

f_R: CCFのナイキスト周波数, $f_R=\min(f_x, f_y)$
f_x, f_y: X と Y のナイキスト周波数

理論的な分析 – 計算量

details

Naive solution
- メモリ空間 $O(n)$
- 時間 $O(n)$

BRAID
- メモリ空間 $O(\log n)$
- 統計値の更新のための時間 $O(1)$
- 補間のための時間 $O(\log n)$
（アウトプットを要求されたとき）

アウトライン

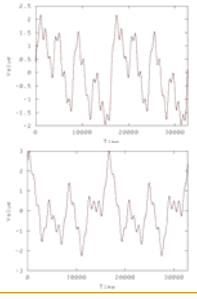
- 研究目的，関連研究
- 前提条件
- 提案手法
- 理論的な分析
- 実験結果

実験結果

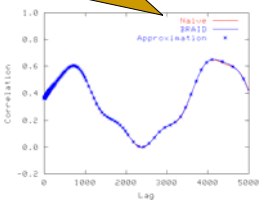
- 設定
  - Intel Xeon 2.8GHz, 1GB memory, Linux
  - データ
    人工データ *Sines*, *SpikeTrains*,
    実データ *Humidity*, *Light*, *Temperature*, *Kursk*, *Sunspots*
  - Enhanced BRAID, b=16

CCFの近似（1）

- *Sines*



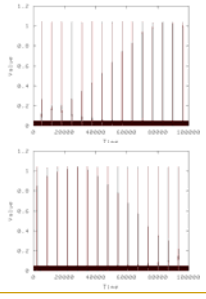
BRAID は，遅延係数を高精度に近似



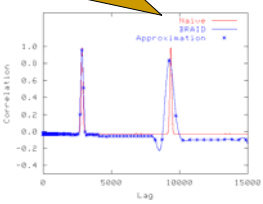
CCF (Cross-Correlation Function)

CCFの近似（2）

- *SpikeTrains*



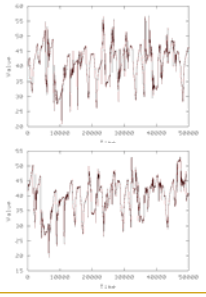
BRAID は，遅延係数を高精度に近似

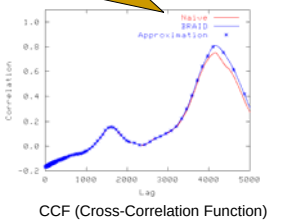


CCF (Cross-Correlation Function)

CCFの近似(3)

- *Humidity* (実データ)

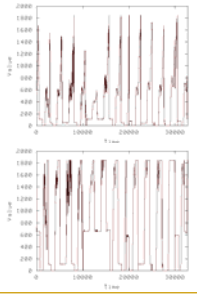


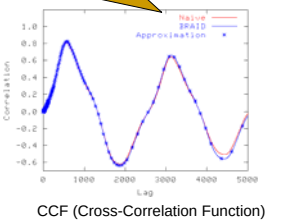


CCF (Cross-Correlation Function)

CCFの近似(4)

- *Light* (実データ)

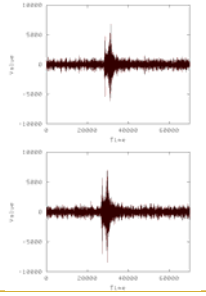


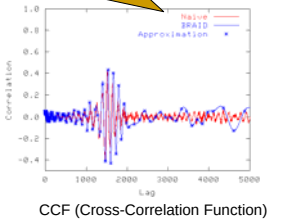


CCF (Cross-Correlation Function)

CCFの近似(5)

- *Kursk* (実データ)

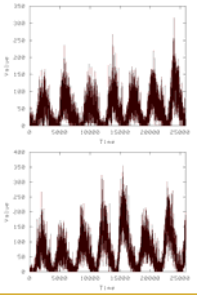


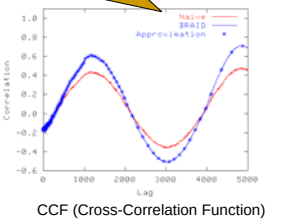


CCF (Cross-Correlation Function)

CCFの近似(6)

- *Sunspots* (実データ)





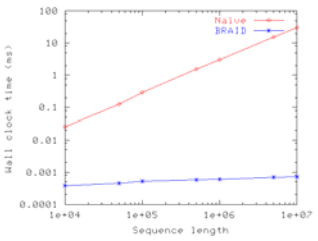
CCF (Cross-Correlation Function)

遅延相関の近似誤差

Datasets	Lag correlation		Estimation error (%)
	Naive	BRAID	
Sines	716	716	0.000
SpikeTrains	2841	2830	0.387
Humidity	3842	3855	0.338
Light	567	570	0.529
Kursk	1463	1472	0.615
Sunspots	1156	1168	1.038

- 相対誤差の最大値は約1%

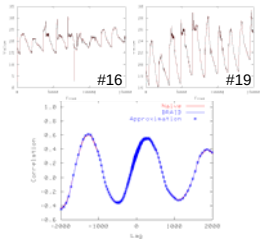
計算時間



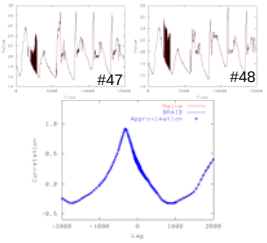
- 大幅に計算時間を削減
- 最大で 約40,000 倍の高速化

複数ペアの遅延相関

- *Temperature*, シーケンス数55
- 相関のあるニつのペア



Estimation of CCF of #16 and #19



Estimation of CCF of #47 and #48

97

理論的な分析 – 精度

details

- サンプリングの効果
  - 概要：標本化定理（ナイキスト）を満たすとき，BRAIDは誤差を生まない
  - 詳細：

以下を満たすとき，BRAIDは必ず遅延相関を見つけることができる

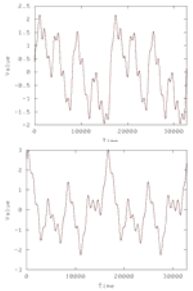
$$0 \le l \le \frac{2b}{f_R}$$

f_R: CCFのナイキスト周波数, $f_R=\min(f_x, f_y)$
f_x, f_y: X と Y のナイキスト周波数

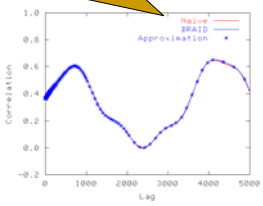
98

CCFの近似（1）

- *Sines*



BRAID は，遅延係数を高精度に近似



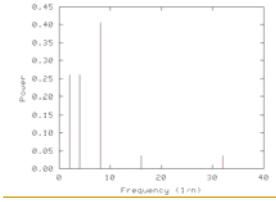
CCF (Cross-Correlation Function)

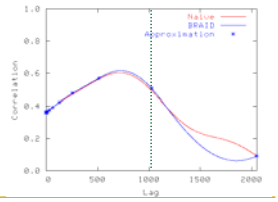
99

サンプリングの効果

details

- データ: *Sines*
- 基本スキーム b=1
- l_R=1024

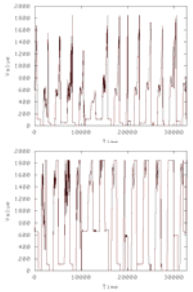




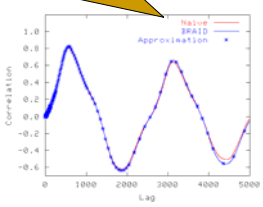
100

CCFの近似（4）

- *Light* (実データ)



BRAID は，遅延係数を高精度に近似



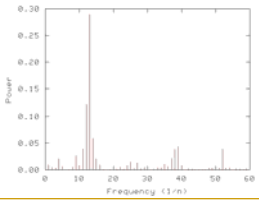
CCF (Cross-Correlation Function)

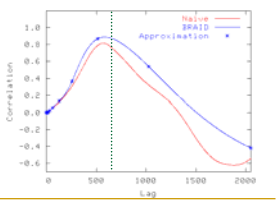
101

サンプリングの効果

details

- データ: *Light*
- 基本スキーム b=1
- l_R=630





102

まとめ

- データストリームのための遅延相関検出

1. いつでも処理可能
2. 省メモリ
   - 統計情報の更新に
     $O(\log n)$ のメモリ空間，$O(1)$ の時間
3. 高速
   - ナイーブな実装と比べて，40,000 倍の高速化
4. 高精度
   - 相対誤差の最大値は約1%

103