Relational Nonlinear FIR Filters

Ronald K. Pearson

Daniel Baugh Institute for Functional Genomics and Computational Biology
Thomas Jefferson University
Philadelphia, PA

Moncef Gabbouj

Institute of Signal Processing
Tampere University of Technology
Tampere, Finland

DIMACS Workshop on Data Quality, Data Cleaning and Treatment of Noisy Data

November 3-4, 2003

Topics

1. A very brief description of the problem
2. Nonlinear FIR filters for data cleaning
3. The relational data model
4. Relational sequence filters
5. Relational region filters
6. Relational cluster filters
7. Summary

What Is the Problem?

- Dasu and Johnson, 2003:

  > Take NOTHING for granted. The data are never what they are supposed to be, even after they are "cleaned up." The schemas, layout, content, and nature of content are never completely known or documented and continue to change dynamically.

- Mendelzon and Mihaila (2001) note that consistency problems are particularly common in applications that combine data from multiple sources

- CAMDA: microarray data analysis contest
  - CAMDA 2002: 2 challenge datasets
  - normal mouse dataset: 3 tissue-specific datasets, constructed from 72 microarrays

  $\leadsto$ data transfer error caused $1,932$ of $5,304$ genes to be mis-indexed in one dataset

Nonlinear FIR Filters

- General notion:
  - given an input sequence $\{x_k\}$
  - desire a related sequence $\{y_k\} = \mathcal{F}\{x_k\}$
  - $\rightsquigarrow$ motivation: noise reduction, outlier elimination, trend removal, feature detection, etc.
- The NFIR solution:

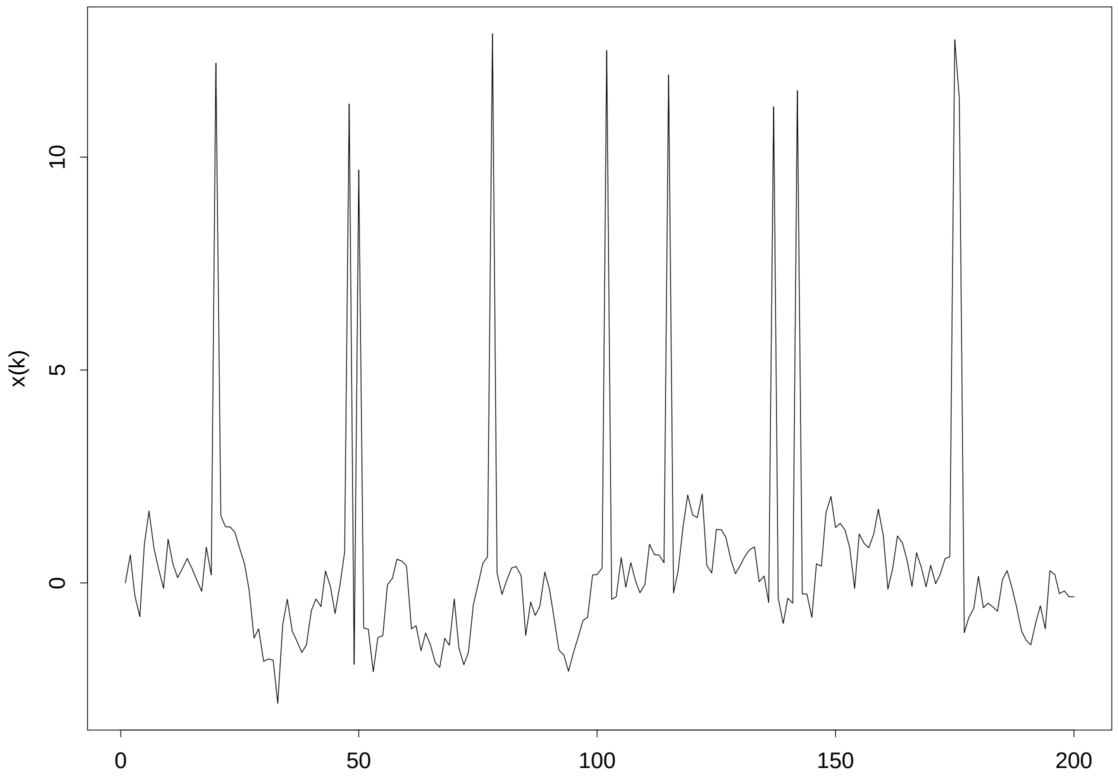
$$y_k = \Phi(x_{k-m}, \ldots, x_{k-n})$$

- Some important examples:
  - linear Finite Impulse Response (FIR) filters (e.g., weighted averages)
  - $\rightsquigarrow$ effective in many applications, but *not* in outlier elimination
  - the standard median filter:

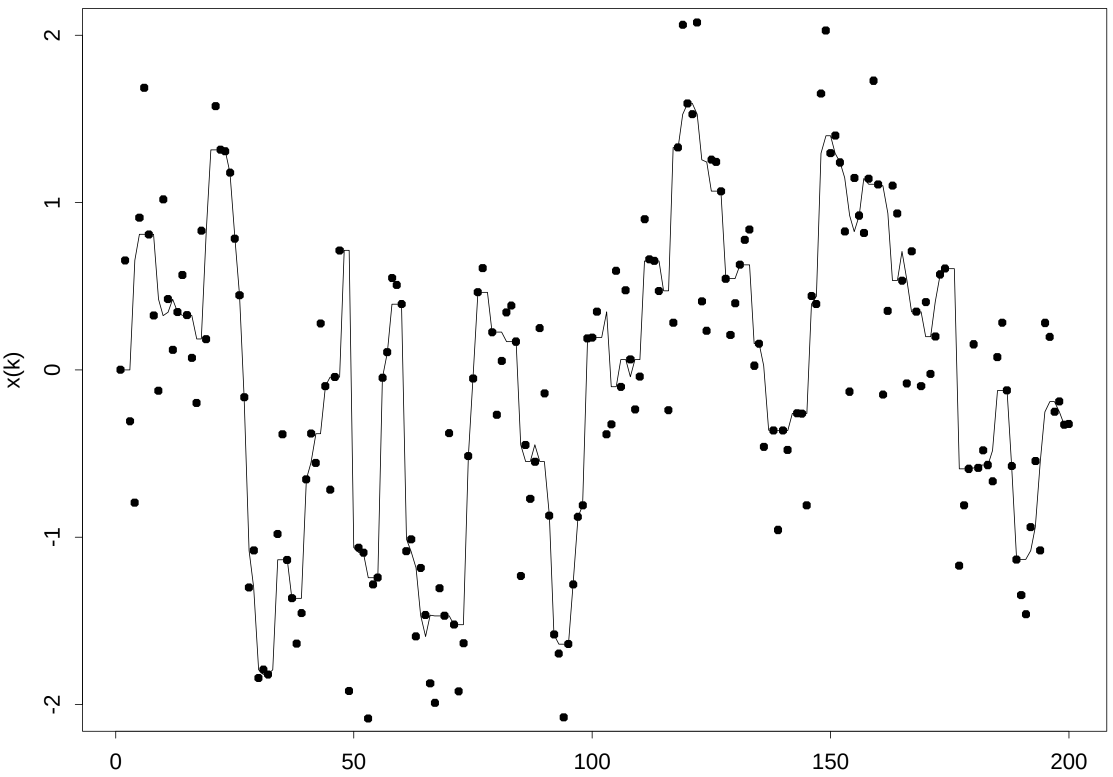
$$y_k = \text{median}\,\{x_{k-K}, \ldots, x_k, \ldots, x_{k+K}\}$$

  - $\rightsquigarrow$ rejects outliers, but usually introduces unacceptable distortion

A Contaminated Data Sequence



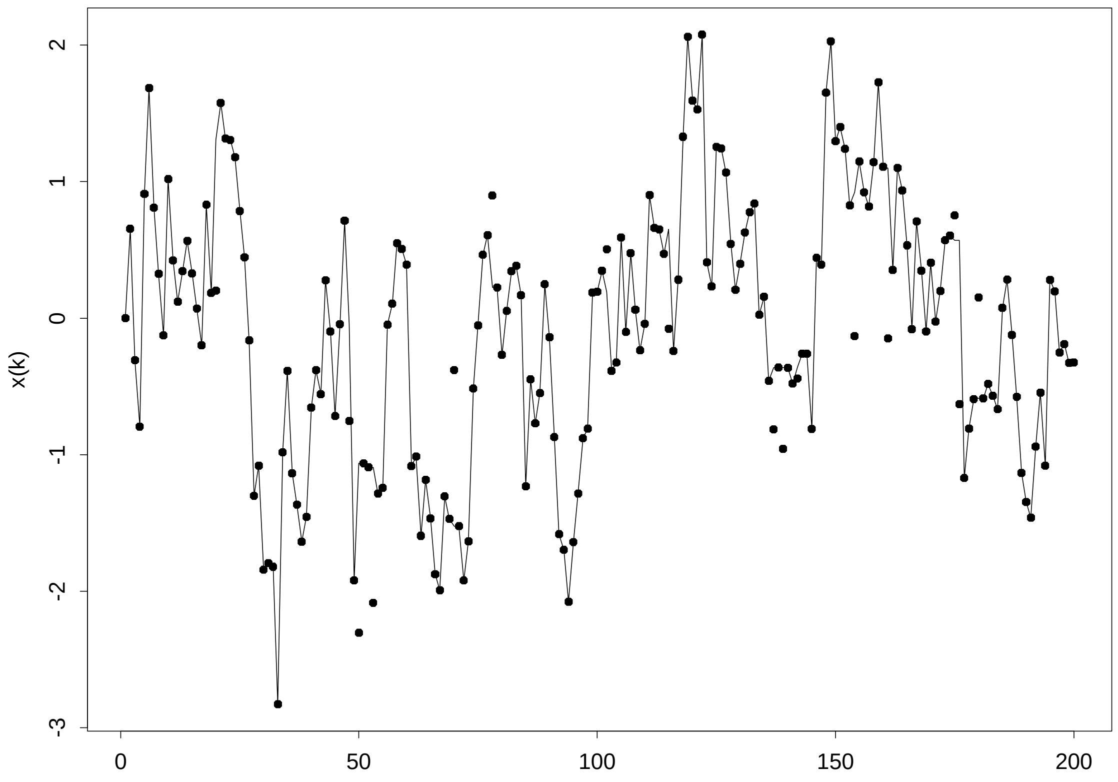
Median Filter Response
$K = 2$



The Hampel Filter

- Basic idea:
  1. form moving data window:
     $$\mathbf{w}_k = \{x_{k-K}, \ldots, x_k, \ldots, x_{k+K}\}$$
  2. test: is x_k an outlier in $\mathbf{w}_k$?
     a. yes $\Rightarrow$ replace with median of $\mathbf{w}_k$
     b. no $\Rightarrow$ make no change
- Outlier detection:
  1. compute MAD scale estimate for $\mathbf{w}_k$
     $$S_k = 1.4826 \text{ median } \{|x_k - \text{median } \{\mathbf{w}_k\}|\}$$
  2. test: $|x_k - \text{median } \{\mathbf{w}_k\}| > tS_k \Rightarrow$ outlier
- Tuning parameters:
  - K = moving window width $\sim$ "bandwidth" (esp., patchy outliers)
  - t = threshold $\Rightarrow$ "aggressiveness"
    * $t = 0 \Rightarrow$ median filter (most aggressive)
    * $t \rightarrow \infty \Rightarrow$ no filtering* (least aggressive)

Hampel Filter Response
$K = 3,\ t = 3$



The Relational Data Model

- Basic structure:
  - relation $\simeq$ data table
  - tuple $\simeq$ table row
  - attribute $\simeq$ table column
- Duplications forbidden:
  - no duplicate tuples (rows)
  - no duplicate attributes (columns)
  - $\rightsquigarrow$ practical problem: duplicates and near-duplicates exist in real databases
- Relational operators preserve tables
- Example: INTERSECT
  - $A, B =$ data tables
  - $A \cap B =$ table containing common tuples from A and B
  - $\Rightarrow$ A and B must have tuples of the same types (i.e., must have the same attributes)

Keys

- Basic idea:

  keys uniquely identify tuples

- More specifically,
  1. a key K is a collection of attributes
  2. $t_1(K) \neq t_2(K)$ for any distinct tuples t_1, t_2
  3. condition 2 fails for any proper subset of K

- Practical issues:
  - keys must be distinct

  $\rightsquigarrow$ in practice, they may not be
  - missing keys are not permitted

  $\rightsquigarrow$ in practice, keys are sometimes missing

$\rightsquigarrow$ Dasu and Johnson (2003), p. 174:

> In many cases, we have been given datasets in which the field which is supposed to be the primary key was not actually unique in all records, but we were able to find other keys in the table and use them to join tables or to verify approximate joins.

The Join Operation

- Basic idea:

  combine two tables with one or more common attributes

- Illustrative example:
  - Table A has attributes $\{X_1, \ldots, X_n, Y_1, \ldots, Y_m\}$
  - Table B has attributes $\{X_1, \ldots, X_n, Z_1, \ldots, Z_p\}$
  - $A \bowtie B = A$ JOIN B has attributes $\{X_1, \ldots, X_n, Y_1, \ldots, Y_m, Z_1, \ldots, Z_p\}$
  - tuples in $A \bowtie B$ are those with the same values for attributes $X_1, \ldots, X_n$ in Tables A and B

- Important operation in practice:

  intended basis for the CAMDA normal mouse dataset: each organ-specific dataset *should* have contained the joins of all 24 individual microarray datasets

Relational Sequence Filters

- Basic idea:

  extend NFIR filter class from real-valued sequences $\{x_k\}$ to sequences $\{T_k\}$ of data tables

- Motivations:

  1. relational data sequences are becoming increasingly important , e.g.
     - $*$ stock market price data
     - $*$ meterological event data
     - $*$ customer purchase pattern data

  $\rightsquigarrow$ Sadri *et al.* (2001): SQL-TS = "Simple Query Language for Time Series"

  2. sequential case represents simplest extension of scalar NFIR filters

$\rightsquigarrow$ **Key NFIR constraint:**

  filter output $\{Y_k\}$ should be a sequence of *valid* data tables, computed from $\{T_{k-K}, \ldots, T_k, \ldots, T_{k+K}\}$

Two Practical Details

1. Key validation: what do we do about missing (i.e., null) or duplicate keys?
   a. strong form: halt processing
   b. semi-strong form: extract the valid subsequence $\{T'_k\}$ of $\{T_k\}$ for subsequent processing

   $\Rightarrow$ *filter must be able to process incomplete data windows*

   c. weak form: create new, unique keys for each tuple in any table in $\{T_k\}$ with a missing or duplicate key

2. Sequence extension: what about end effects?

   $\leadsto$ $\{T_{k-K}, \ldots, T_{k+K}\}$ not all defined for $1 \leq k < K$ or $N - K < k \leq N$

   – extension strategy (J.W. Tukey): define $T_j = T_1$ for $j = -K+1, \ldots, 0$ and $T_j = T_N$ for $j = N+1, \ldots, N+K$

Two Filtering Strategies

$\rightsquigarrow$ Basic view:

represent a table T_k with p tuples and q attributes as a $p \times q$ matrix

1. Diagonal filtering strategy:

$$Y_k^{ij} = \Phi_{ij}(T_{k-K}^{ij}, \ldots, T_k^{ij}, \ldots, T_{k+K}^{ij})$$

   - equivalent to pq scalar NFIR filters
   - advantage: simplicity
   - disadvantage: takes no advantage of any useful intra-table data dependences

2. Full filtering strategy:

$$\mathbf{Y}_k = \Phi(\mathbf{T}_{k-K}, \ldots, \mathbf{T}_k, \ldots, \mathbf{T}_{k+K})$$

   - equivalent to a MIMO filter with pq inputs and pq outputs
   - advantage: can take full advantage of useful intra-table dependences
   - disadvantage: complexity

A Third Strategy:

Block-structured Filtering

- Basic idea:

  partition data tables $\mathbf{T}_k$ into smaller subtables $\mathbf{T}_k^\ell$ of related variables

$\leadsto$ Note that blocks need not be disjoint:

  $\mathbf{T}_k^\ell$ and $\mathbf{T}_k^m$ can share variables for $m \neq \ell$ but should not be identical

- Implementation:

$$\mathbf{Y}_k^\ell = \Phi_\ell(\mathbf{T}_{k-K}^\ell, \ldots, \mathbf{T}_k^\ell, \ldots, \mathbf{T}_{k+K}^\ell)$$

- Equivalent to applying a MIMO filter of the appropriate dimension to each subtable $\mathbf{T}_k^\ell$
- Advantage: can be tailored to take advantage of the most useful intra-table dependences
- Disadvantage: more complex to implement than the diagonal strategy, but simpler than the full strategy

Domain Restrictions

- Traditional NFIR filters map real numbers into real numbers

$\rightsquigarrow$ Relational data tables often contain other data types, e.,g. integers, category designations, character strings

$\Rightarrow$ Data table validity requirements may impose significant restrictions on the function $\Phi(\cdot)$

- consequence: certain "standard" operations may be forbidden
- example: x_k must be an integer

$$\begin{aligned}\Phi(\mathbf{w}_k) &= \frac{1}{2K+1}\sum_{j=-K}^{K} x_k \\ &\Rightarrow \text{averages inadmissable} \\ \Phi(\mathbf{w}_k) &= \text{median}\,\{x_{k-K},\ldots,x_{k+K}\} \\ &\Rightarrow \text{odd medians admissable}\end{aligned}$$

$\rightsquigarrow$ Particular challenge: block-structured filtering with mixed data types

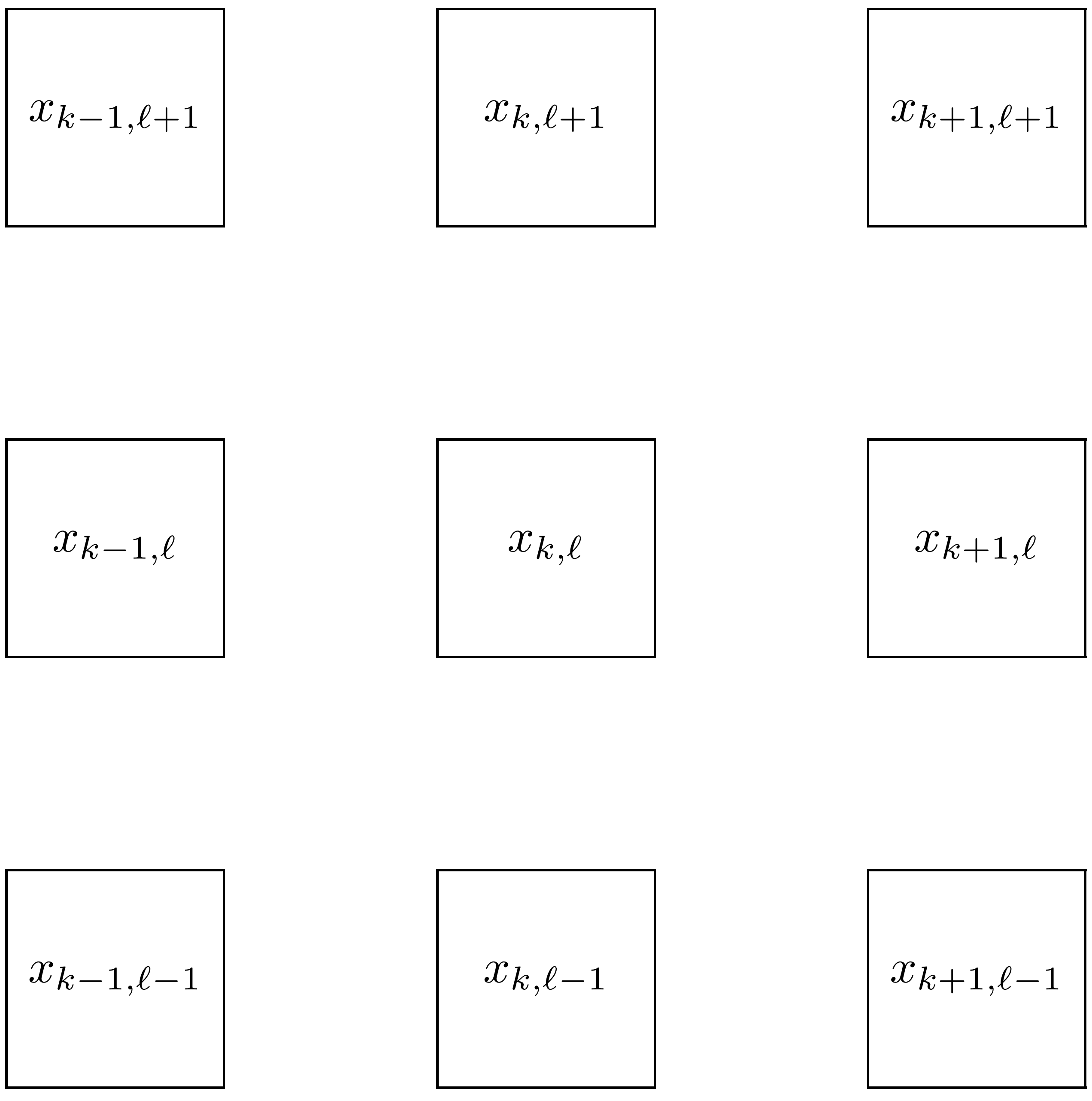
Relational Region Filters

- Motivation: image processing
  - replace $1D$ sequence with $2D$ sequence
  - i.e., $\{x_{k-K}, \ldots, x_k, \ldots, x_{k+K}\} \rightarrow \{x_{k-K,\ell-L}, \ldots, x_{k,\ell}, \ldots, x_{k+K,\ell+L}\}$
- Relational region filter:
  - replace table sequence $\{T_k\}$ with multiply-indexed sequence
  - example: sales data tables $\{T_{tksz}\}$
    - $t =$ product type
    - $k =$ day or week of transaction
    - $s =$ store location
    - $z =$ customer zipcode

$\rightarrow$ Two new issues:

1. total vs. partial orders
2. window complexity

A 3×3 Window Example



Total vs. Partial Orders

- Total order: for every i and j
  1. i preceeds $j \Leftrightarrow i < j$
  2. i follows $j \Leftrightarrow i > j$
  3. i is the same as $j \Leftrightarrow i = j$
- Fourth possibility for a partial order:
  4. i neither preceeds nor follows j and $i \neq j$

$\leadsto$ In a total order, $i < j < k \Rightarrow T_i$ is further from T_k than T_j

  – useful, e.g., downweight “remote” tables

  – does not extend to partial orders

$\leadsto$ For a one–dimensional sequence $\{T_k\}$, k defines a total order, but for multi-dimensional sequences like $\{T_{tksz}\}$, indices only define a partial order

$\leadsto$ e.g., does $x_{k,\ell+1}$ preceed or follow $x_{k+1,\ell}$?

Window Complexity Issues

- Size depends strongly on index dimension
  - 1D, symmetric window: $S = 2K + 1$
  - 2D, square window: $S = (2K + 1)^2$
  - general nD, full window: $S = (2K + 1)^n$
- Some representative full window sizes:

K	1D	2D	3D	4D	5D
1	3	9	27	81	243
2	5	25	125	625	3,125
3	7	49	343	2,401	16,807
4	9	81	729	6,561	59,049
5	11	121	1,331	14,641	161,051

Two Practical Compromises

- Compromise 1:

  impose structure to reduce size

- Image processing examples:
  - full square window: $S = (2K+1)^2$
  - cross-shaped window: $S = 4K+1$
  - X-shaped window: $S = 4K+1$
- General extensions:
  - full hypercube window: $S = (2K+1)^n$
  - cross-shaped window:
    $S = (2K+1) + (n-1) \cdot 2K = 2nK + 1$
  - X-shaped window: $S = 2nK + 1$
- Compromise 2:

  combine 1D subwindows using composite filters

- 2D example—separable median filter:

$$\begin{aligned} y_{k,\ell} \quad = \quad & \text{median}\,\{\text{median}\,\{\mathbf{w}^{D}_{k,\ell}\}, \\ & \quad \text{median}\,\{\mathbf{w}^{H}_{k,\ell}\}, \text{median}\,\{\mathbf{w}^{V}_{k,\ell}\}\} \end{aligned}$$

Relational Cluster Filters

- Basic idea:
  - relational region filters are based on indices of adjacent records
  - $\rightsquigarrow$ underlying assumption for data cleaning: adjacent records should be similar
  - $\rightarrow$ alternative measures of record similarity?
- Related idea: cluster analysis
  - basic problem: partition a set $\mathcal{S}$ of objects into k disjoint subsets
  - clustering algorithms typically based on dissimilarities between objects
- Specific extension for relational filters:
  - for each data table T_k, select its $2K$ most similar neighbors
  - apply a relational sequence filter based on T_k and its $2K$ most similar neighbors
  - $\rightsquigarrow$ can these neighbors be ordered? If not, use permutation-invariant filters

Dissimilarity Measures

- Basic idea: d_{ij} measures the degree of dissimilarity between objects i and j
- Standard requirements:
  - nonnegativity: $d_{ij} \geq 0$
  - self-similarity: $d_{ii} = 0$
  - symmetry: $d_{ij} = d_{ji}$
- Examples:
  - Euclidean distance:

$$d_{ij} = \sqrt{\sum_{k=1}^{n} (x_k^i - x_k^j)^2}$$

$\rightsquigarrow$ any norm defines a dissimilarity

$\rightsquigarrow$ different data types require alternatives

  - e.g., Jaccard measure for binary data:

$$\begin{aligned} d_{ij} &= \frac{b+c}{a+b+c} \\ a &= |\{\, i,j \mid x_i = x_j = 1\}| \\ b &= |\{\, i,j \mid x_i = 1, x_j = 0\}| \\ c &= |\{\, i,j \mid x_i = 0, x_j = 1\}| \end{aligned}$$

Relational Dissimilarities

- Attributes can be of many different types
- Some simple combined measures have been proposed (Gordon, 1999)

$\rightsquigarrow$ An interesting extension:

- compute dissimilarity matrices Δ_ℓ for individual attributes or related subsets
- combine $\{\Delta_\ell\}$ to form Δ
- possible combinations:
  - sum rule:

$$\Delta = \sum_{\ell=1}^{r} \Delta_\ell$$

  - Haddamard product:

$$\Delta_{ij} = \prod_{\ell=1}^{r} \Delta_{ij}^{\ell}$$

$\rightsquigarrow$ must preserve dissimilarity properties

$\Rightarrow$ standard matrix products excluded

Approximate Join Filters

- Problem:
  - Table A contains part of the information
  - Table B contains the rest
  - $\Rightarrow$ desired information is in the join $A \bowtie B$
  - $\leadsto$ tables are incomplete/corrupted
- Statistical file merging (Barr & Turner, 1990):
  - Table $A = \{X_1, \ldots, X_n, Y_1, \ldots, Y_m\}$
  - Table $B = \{X'_1, \ldots, X'_n, Z_1, \ldots, Z_p\}$
  - exact match records for which $X'_i = X_i$ generally do not exist
  - $\leadsto$ alternative: match records for which X'_i is as similar to X_i as possible and form the *approximate join*: $A \diamond B = \{X_1, \ldots, X_n, Y_1, \ldots, Y_m, Z'_1, \ldots, Z'_p\}$
- Approximate join cluster filter:
  $A \circ B = \{X_1, \ldots, X_n, Y_1, \ldots, Y_m, \hat{Z}_1, \ldots, \hat{Z}_p\}$
  where $\hat{Z}_i$ is obtained from a cluster filter based on $2K$ nearest neighbor tuples from B

Summary

- Preliminary overview of what appears to be a promising research area
- Key ideas:
  - extend nonrecursive NFIR filter class to relational data objects
  - use *local neighborhood information*
- Key issues:

  $\rightsquigarrow$ how do we define local neighborhoods?

  $\rightsquigarrow$ how do we combine different data types?

  $\rightsquigarrow$ how do we manage window complexity?
- A useful extension:

  nonlinear FIR filters for cleaning semi-structured data objects (e.g., XML documents)