Streaming Algorithms II
Preprocessing and Transformation

Objectives

- Feedback on Seminars
- Projects
- Brief Recap
- Streaming Algorithms for Pre-processing and Transformation
  - Transforms
  - Dimensionality Reduction
- Intro to Modeling

Aside: Projects

- https://sites.google.com/site/fundamentalsofstreamprocessing/projects

- Data/Input
  - What data sources will you use?
  - Will the data be stored and replayed, or pulled in live?
  - Do you need to create new connectors to access the data?
- Techniques/Application
  - What is the problem you will solve?
  - Is there a set of references that motivate this?
  - Do you need to integrate other tools?
- Results
  - How will you present your results?
  - What will you show as a demo?
  - What techniques will you plan to use?
- Next Steps
  - Will you use any algorithms from class? Will you use any optimizations

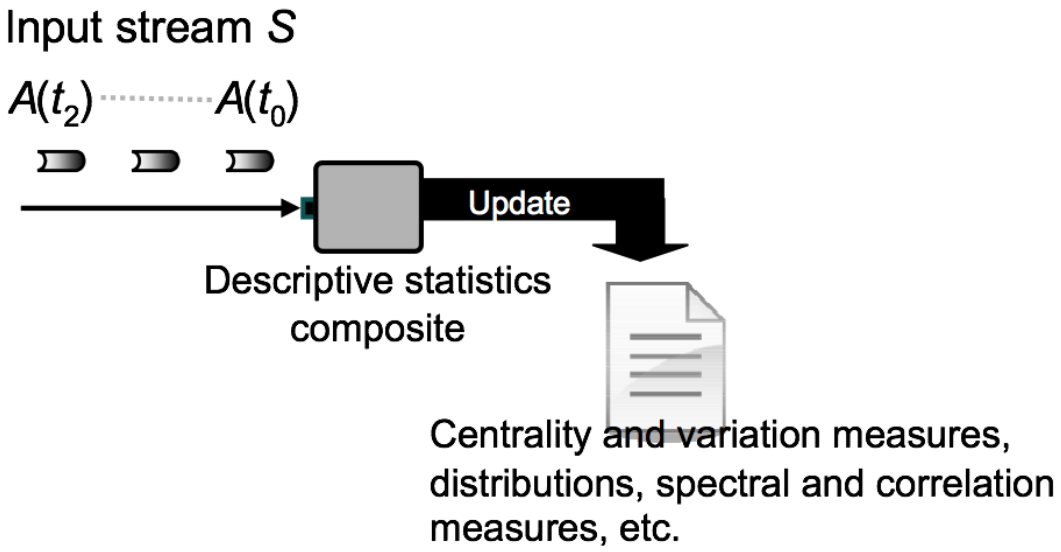
Recap: Outline of Data Mining Process

- Data acquisition
  - Collect data from external sources
- Data pre-processing
  - Prepare data for further analysis: data cleaning, data interpolation (missing data), data normalization (heterogeneous sources), temporal alignment and data formatting, data reduction
- Data transformation
  - Select an appropriate representation for the data
  - Select or compute the relevant features (*feature extraction/selection*)
- Modeling (data mining)
  - Identify interesting patterns, similarity and groupings, partition into classes, fit functions, find dependencies and correlations, identifying abnormal data
- Evaluation
  - Use of the mining model and evaluation of the results

Recap: Data Preprocessing and Transformation

- Descriptive statistics
  - Extracting simple quantitative statistics of the distribution
- Sampling
  - Reducing the volume of the input data by retaining only an appropriate subset for analysis
- Sketches
  - Compact data structures that contain synopses of the streaming data, for approximate query answering
- Quantization
  - Reduce the fidelity of individual data samples to lower compute and memory costs
- Dimensionality reduction
  - Reduce the number of attributes within each tuple to decrease data volume and improve accuracy of the models
- Transforms
  - Convert data items or tuples and their attributes from one domain to another, such that data is better suited for further analysis

Descriptive Statistics



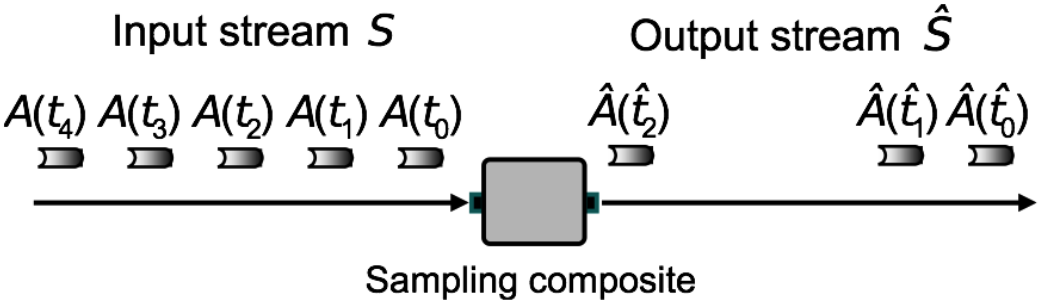
- Extract information from the tuples in a stream such that
  - statistical properties of the stream can be characterized or summarized
  - the quality of the data it transports assessed

Descriptive Stats: BasicCounting

- Question: Assume you have a stream that contains tuples that are either 0s or 1s, how can you maintain the number of 1s in the last W tuples
  - Sliding window of size W, slide 1
- Assume W is large
  - So $O(W)$ is too large. We want to store logarithmic space
- The *BasicCounting* algorithm
  - Uses space of the order $O\left(\frac{1}{\varepsilon}\log^2 W\right)$
  - With error tolerance ε, i.e. the counts are within $1\pm\varepsilon$ of its actual value. ε is specified by application objective
- Study: Data structure, update, query

Sampling

- Sampling is used to reduce the data stream by selecting some tuples based on one or more criteria



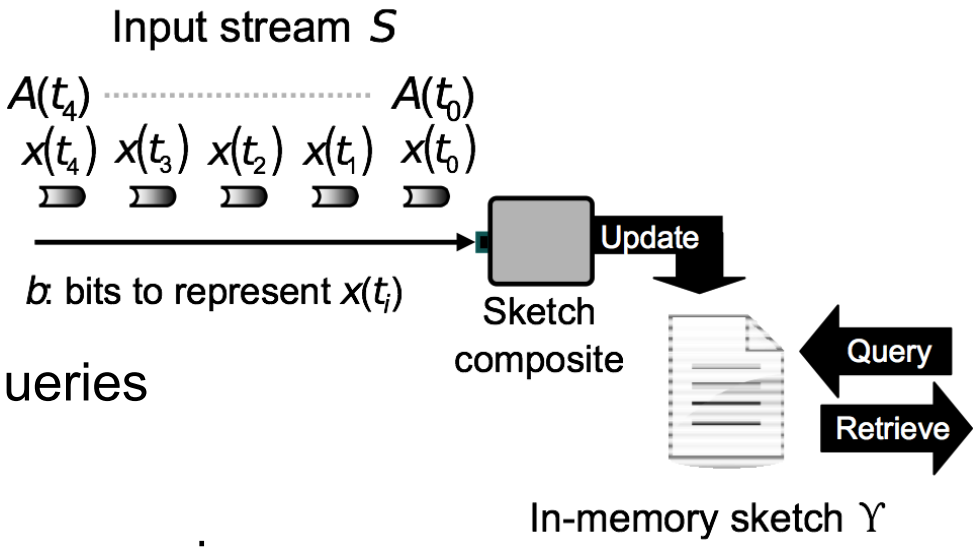
- Three types of sampling
  - Systematic or Uniform
  - Data Driven
  - Random

Reservoir Sampling Algorithm

- Let q be the reservoir size
  - Indices in reservoir run from 0 to q-1
- Let i be the index of the tuple being processed right now
- If $i<q$
  - Append the tuple to the reservoir
- Otherwise
  - Select p as a random integer in range $[0, i]$
  - If $0 \leq p < q$
    - Replace tuple at index p in reservoir with current tuple

Sketches

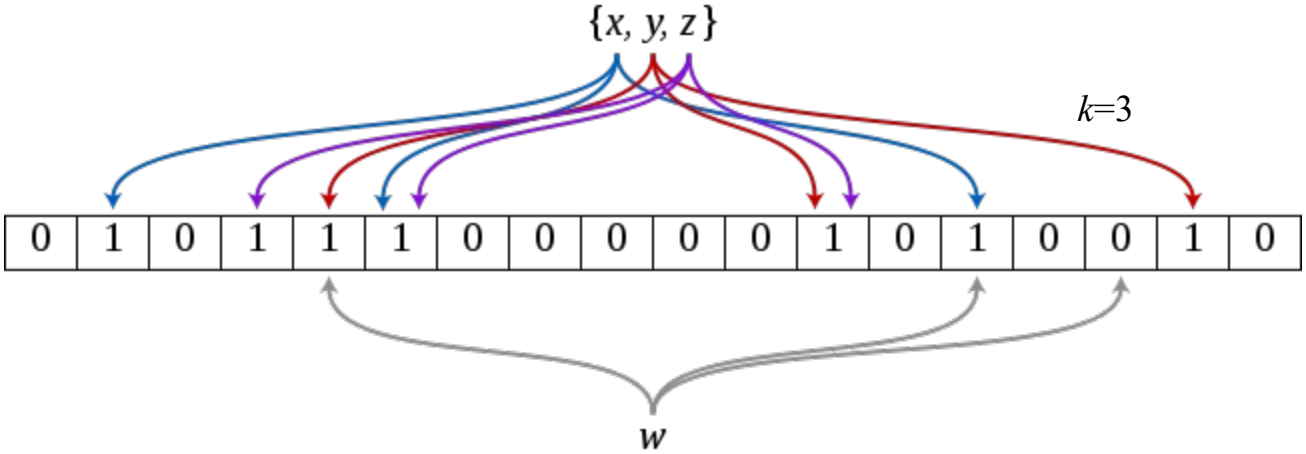
- In-memory data structures that contain compact, often lossy, synopses of streaming data
- They capture the key properties of the stream



- In order to answer specific queries
  - Error bounds and
  - probabilistic guarantees
- Used for approximate query processing

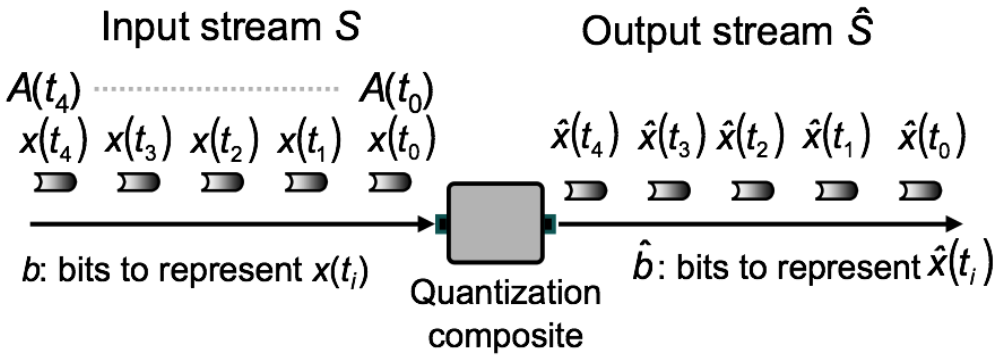
Bloom Filters

- Let us say we want to answer containment queries
- Whether a given item has been see so far or not
- Bloom filter
  - Keep a bit array of size m
  - Use k pairwise independent hash functions (more later)
  - Update: Hash item to k locations and set the bits to one
  - Query: Hash item to k locations
    - Return true if all of the k locations are set

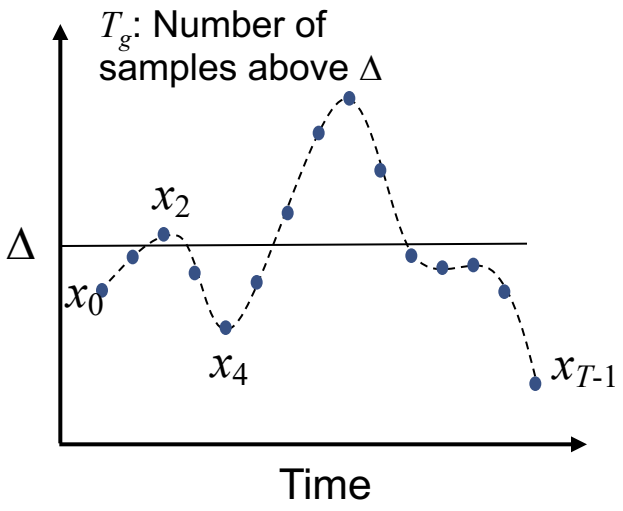


Quantization

- Simplest lossy data reduction mechanism
  - Perfect reconstruction of original signal not possible
- Maps Input Sample → Reconstruction Value from predefined codebook
- Example
  - Rounding Function
  - Floor Function
  - Truncation
- Original sample
  - Numeric Tuples with Continuous or Discrete Space



Moment Preserving Quantization



$$Q(x_i) = \begin{cases} \overbrace{\mu + \sigma\sqrt{\frac{T - T_g}{T_g}}}^{a}; x_i \geq \Delta \\ \underbrace{\mu - \sigma\sqrt{\frac{T_g}{T - T_g}}}_{b}; x_i < \Delta \end{cases}$$

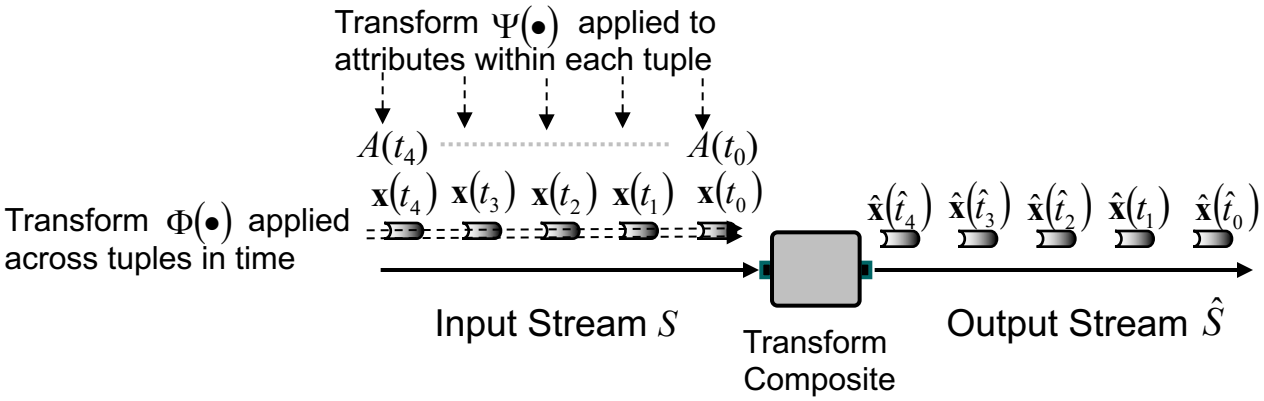
This choice of a and b guarantees preservation of first two moments

$$\frac{1}{T}\sum_{i=0}^{T-1} Q(x_i) = \frac{1}{T}\sum_{i=0}^{T-1} x_i$$

$$\frac{1}{T}\sum_{i=0}^{T-1} [Q(x_i)]^2 = \frac{1}{T}\sum_{i=0}^{T-1} (x_i)^2$$

Transforms

- Transform a tuple's attributes from one domain to another
  - Within tuple transform (for multivariate processing)
  - Across tuple transform (typically for univariate processing)
    - Temporal Transform



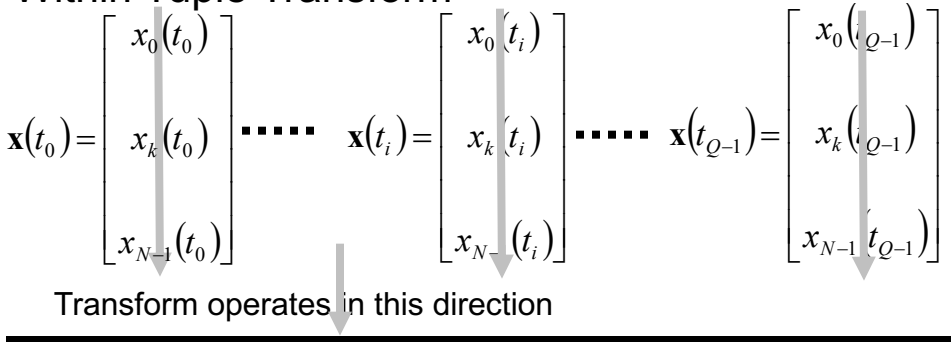
- Focus on Linear Transforms

1-D Transforms

$$\mathbf{x}(t_0) = \begin{bmatrix} x_0(t_0) \\ \vdots \\ x_k(t_0) \\ \vdots \\ x_{N-1}(t_0) \end{bmatrix} \cdots\cdots \mathbf{x}(t_i) = \begin{bmatrix} x_0(t_i) \\ \vdots \\ x_k(t_i) \\ \vdots \\ x_{N-1}(t_i) \end{bmatrix} \cdots\cdots \mathbf{x}(t_{Q-1}) = \begin{bmatrix} x_0(t_{Q-1}) \\ \vdots \\ x_k(t_{Q-1}) \\ \vdots \\ x_{N-1}(t_{Q-1}) \end{bmatrix}$$

Q numeric input tuples with N attributes each

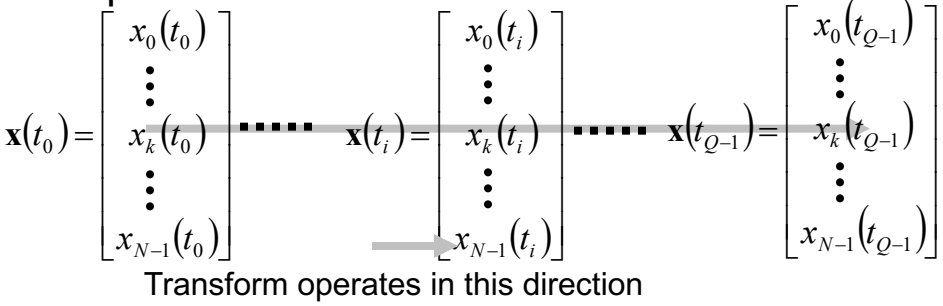
Within Tuple Transform



$$\mathbf{y} = \mathbf{x}(t_i)$$

$$\Psi(\mathbf{y}) = \Lambda_{N\times N}\mathbf{x}(t_i)$$

Temporal Transform



$$y_i = x_k(t_i)$$

$$\Phi(\mathbf{y}) = \Lambda_{Q\times Q}\mathbf{y}$$

Refresh on 1-D Linear Transforms

What makes a transform a linear transform?

Represent signal with N samples as vector

$$\mathbf{y} = [y_0, y_1, \ldots, y_{N-1}]^T$$

1-D transform represents signal as a linear combination of N basis vectors

In general, basis vector may be complex

$$\mathbf{u}_k \in C^N$$

This means $\mathbf{t} \in C^N$ even if $\mathbf{y} \in R^N$

$$\mathbf{y} = \sum_{k=0}^{N-1} t_k \mathbf{u}_k$$

$\mathbf{u}_k$: basis vector (linearly independent)

t_k: transformed coefficients

With $\mathbf{t} = [t_0, t_1, \ldots, t_{N-1}]^T$ and $\mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{N-1}]$ we have

$$\mathbf{y} = \mathbf{U}\mathbf{t}$$ Inverse transform

and

$$\mathbf{t} = \mathbf{U}^{-1}\mathbf{y}$$ Forward transform

Unitary 1-D Linear Transforms

Basis vectors are orthonormal

$$\left\langle \mathbf{u}_j, \mathbf{u}_k \right\rangle = \mathbf{u}_j^H \mathbf{u}_k = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{u}_j^H = \left(\mathbf{u}_j^*\right)^T$ transpose of complex conjugate

The matrix $\mathbf{U}$ is called unitary and we have $\mathbf{U}^H\mathbf{U} = \mathbf{U}\mathbf{U}^H = \mathbf{I}$

$\mathbf{y} = \mathbf{U}\mathbf{t}$ Inverse transform

and

$\mathbf{t} = \mathbf{U}^H\mathbf{y}$ Forward transform

1-D Fourier Transform

$$t_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} y_n e^{-j\frac{2\pi kn}{N}}$$ Forward transform

$$y_n = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} t_k e^{j\frac{2\pi kn}{N}}$$ Inverse transform

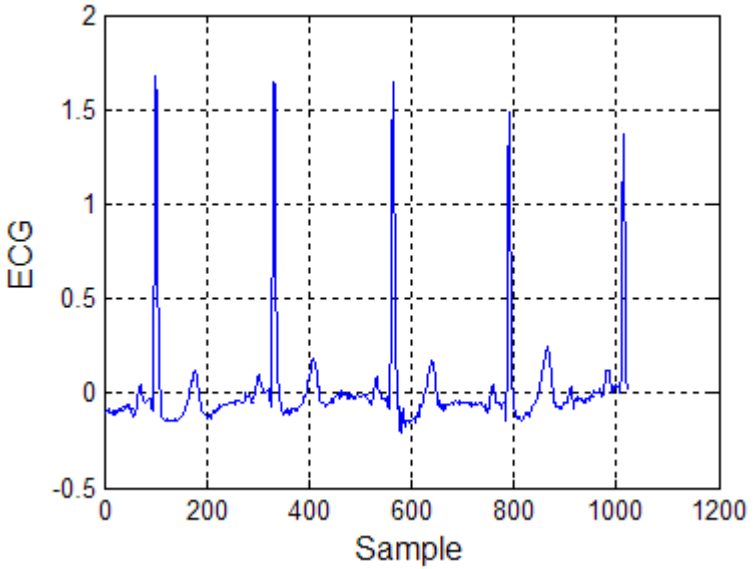
$$\mathbf{u}_k = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 \\ e^{j\frac{2\pi k}{N}} \\ \vdots \\ e^{j\frac{2\pi k(N-1)}{N}} \end{bmatrix}$$

$$\mathbf{U} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{j\frac{2\pi}{N}} & \cdots & e^{j\frac{2\pi(N-1)}{N}} \\ \vdots & \vdots & \cdots & \vdots \\ 1 & e^{j\frac{2\pi(N-1)}{N}} & \cdots & e^{j\frac{2\pi(N-1)(N-1)}{N}} \end{bmatrix}$$

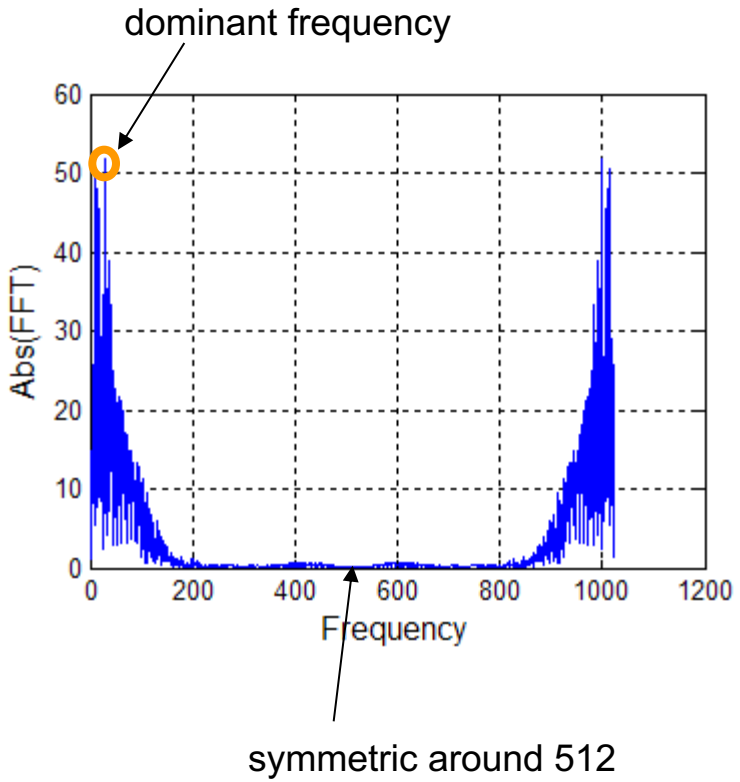
For real signal $\mathbf{y}$ we have $t_k = t^*_{N-k}$ or symmetry in Fourier Transform

1-D Fourier Transform

For real signal $\mathbf{y}$, n has interpretation of time, k has interpretation of frequency



1024 point sampled ECG signal



1-D Discrete Cosine Transform

Similar* to the Fourier transform but uses only real valued bases

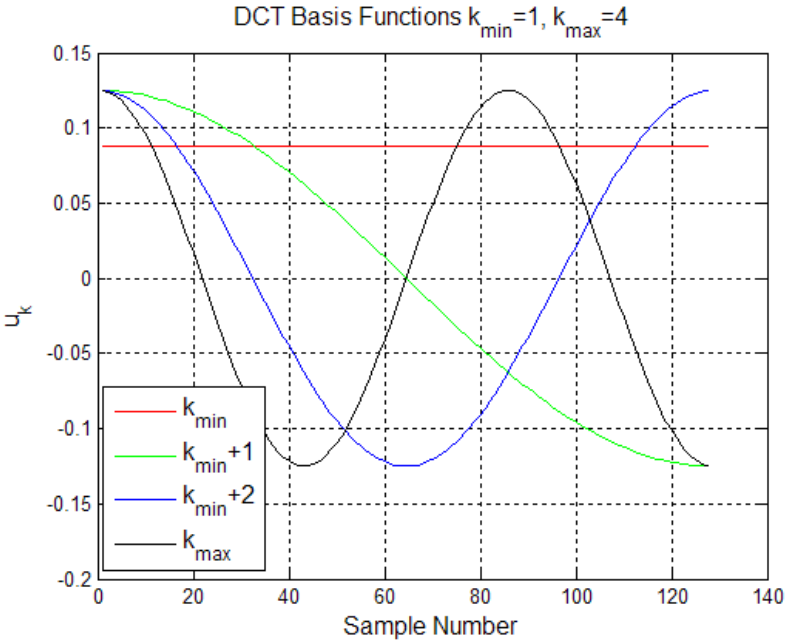
$$t_k = \sum_{n=0}^{N-1} y_n \alpha(k) \cos\left(\frac{(2n+1)k\pi}{2N}\right)$$ Forward transform

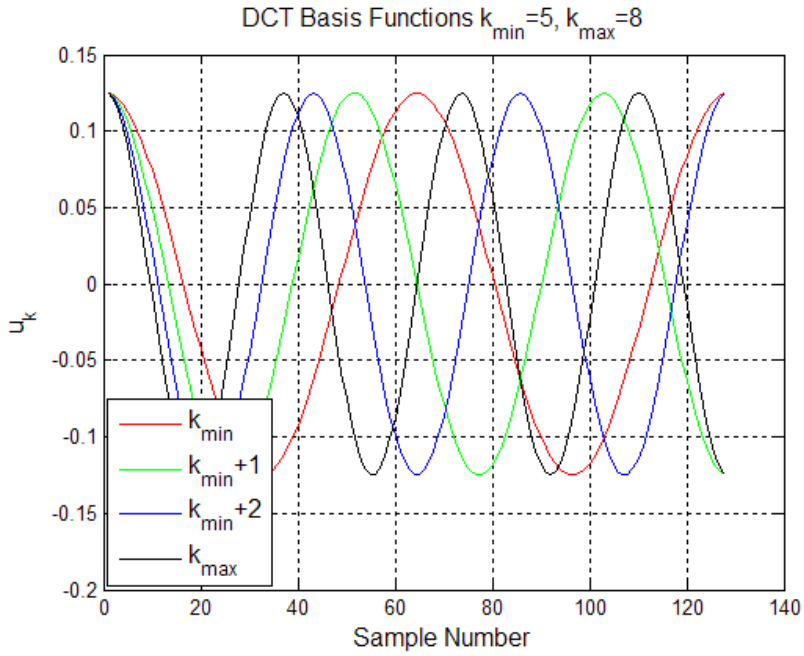
$$\alpha(k) = \begin{cases} \frac{1}{\sqrt{N}}; & k = 0 \\ \sqrt{\frac{2}{N}}; & \text{otherwise} \end{cases}$$

$$y_n = \sum_{k=0}^{N-1} t_k \alpha(k) \cos\left(\frac{(2n+1)k\pi}{2N}\right)$$ Inverse transform

$$\mathbf{U} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & \sqrt{2}\cos\left(\frac{\pi}{2N}\right) & \cdots & \sqrt{2}\cos\left(\frac{(N-1)\pi}{2N}\right) \\ 1 & \sqrt{2}\cos\left(\frac{3\pi}{2N}\right) & \cdots & \sqrt{2}\cos\left(\frac{3(N-1)\pi}{2N}\right) \\ \vdots & \vdots & \cdots & \vdots \\ 1 & \sqrt{2}\cos\left(\frac{(2N-1)\pi}{2N}\right) & \cdots & \sqrt{2}\cos\left(\frac{(2N-1)(N-1)\pi}{2N}\right) \end{bmatrix}$$

1-D Discrete Cosine Transform

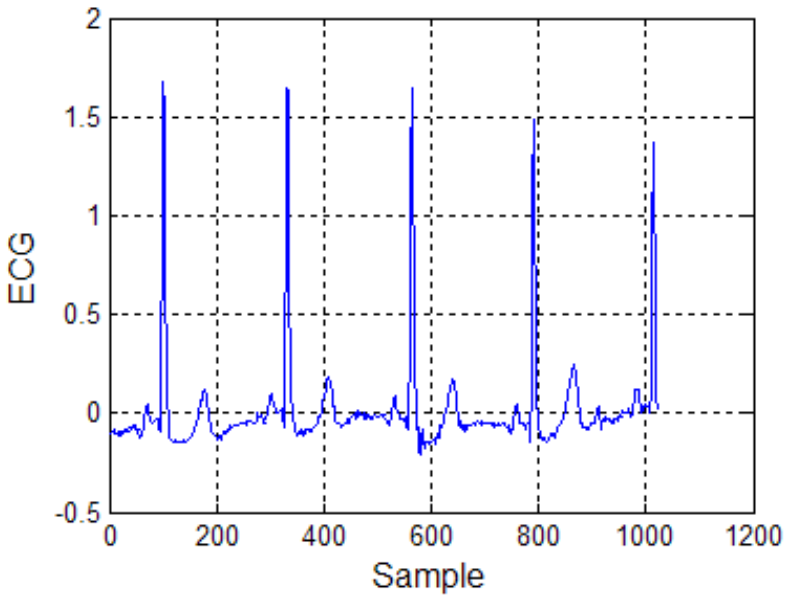




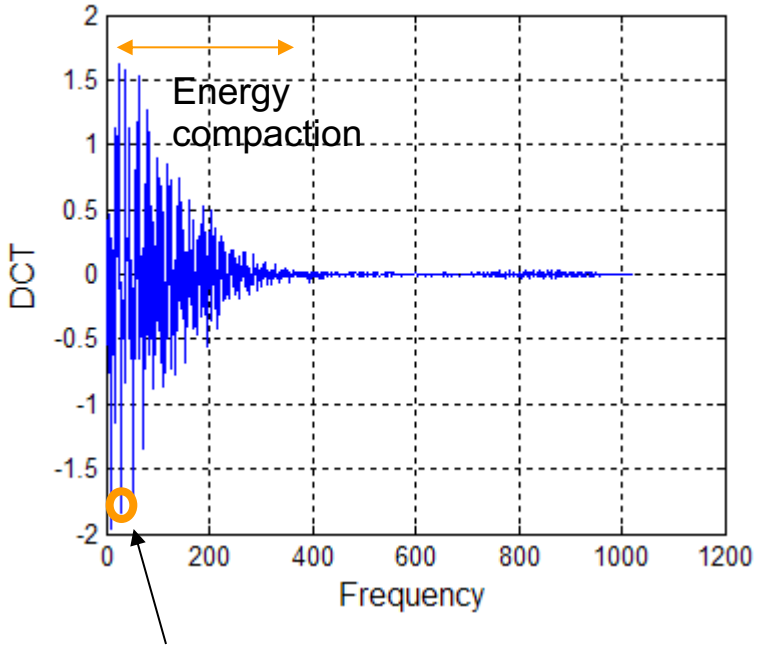
First 8 basis vectors for N=128

Basis functions are sinusoids with increasing frequency

1-D Discrete Cosine Transform

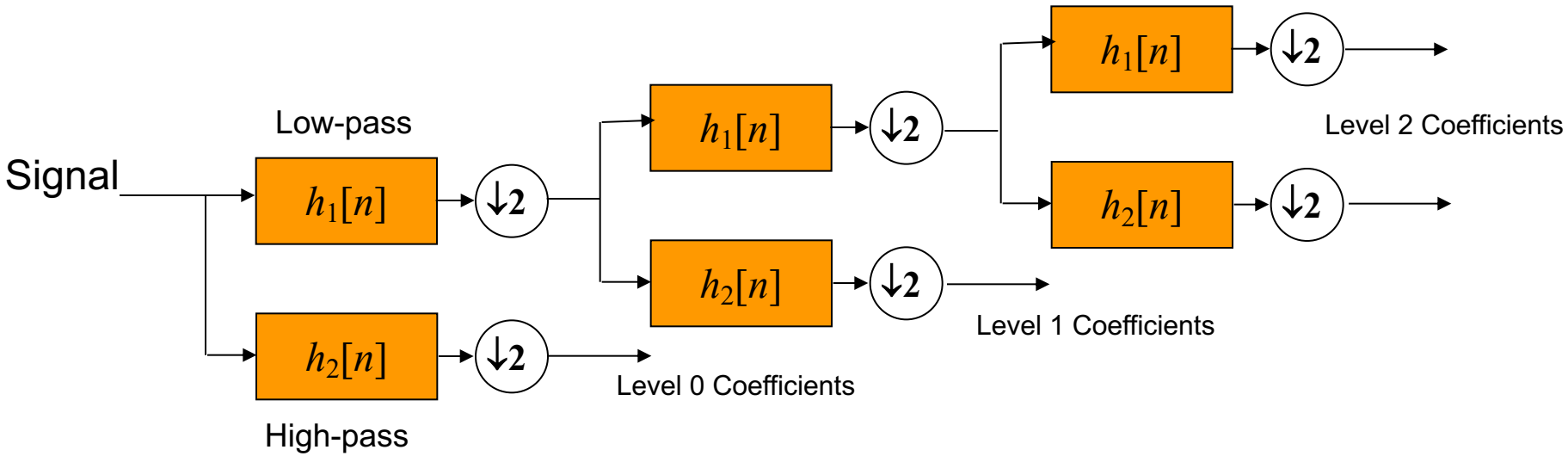


1024 point sampled ECG signal



dominant frequency

Subband Coding: Wavelet Transform

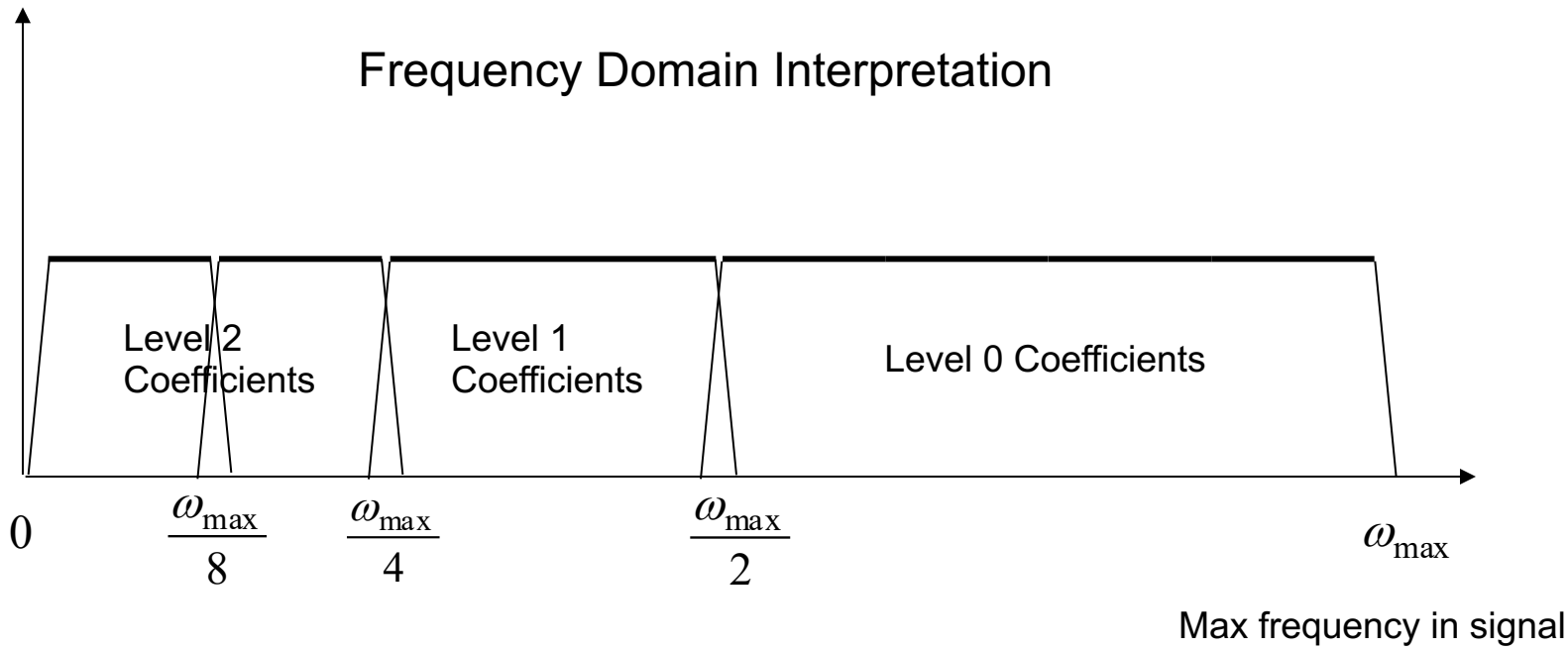


Recursively apply decomposition to low freq band $\Rightarrow$ increase frequency resolution

Set of filters collectively called – "Filter bank"

Multi-resolution representation of signal – **Scalable Coding**

Subband Coding: Wavelet Transform



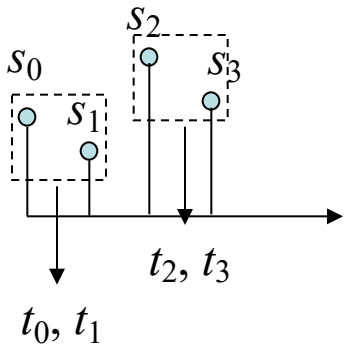
Wavelet Transform: Recursively partitions frequency space
Provides time-frequency localization

1-D Haar Wavelet Transform

$$\mathbf{U} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad \mathbf{U}^H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

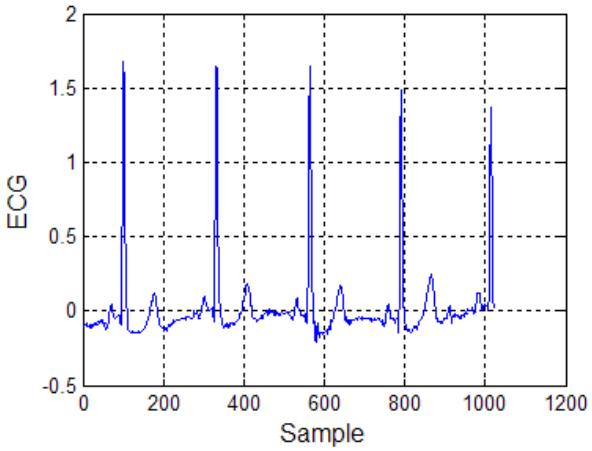
$$t_0 = \frac{s_0 + s_1}{\sqrt{2}} \qquad t_1 = \frac{s_0 - s_1}{\sqrt{2}}$$

Normalized sum and difference

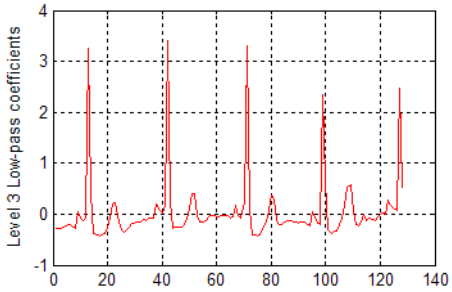


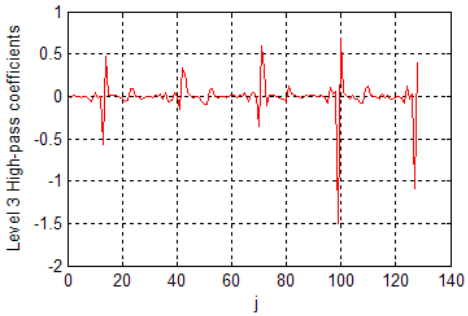
Windowing Based Implementation

1-D Haar Wavelet Transform

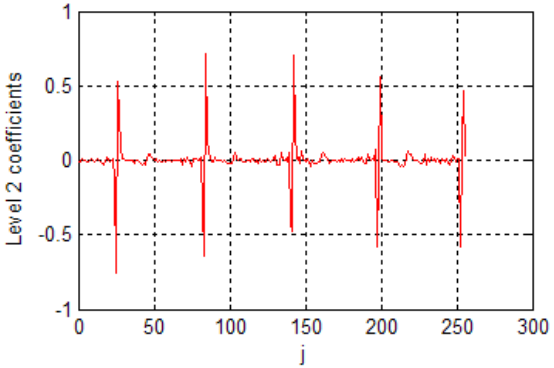


Original Signal

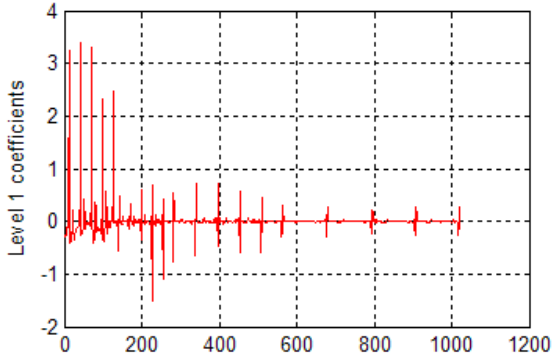




Level 3 Coefficients



Level 2 Coefficients

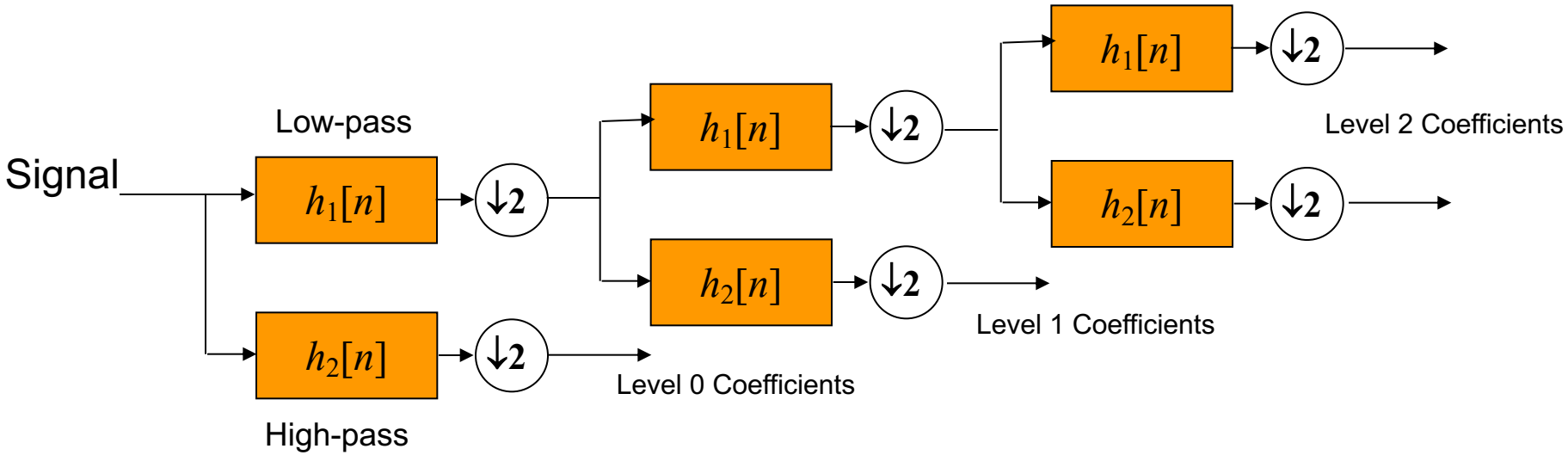


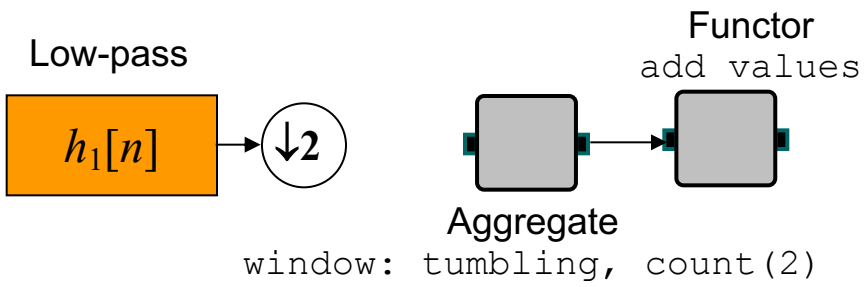
Haar Wavelet Transform

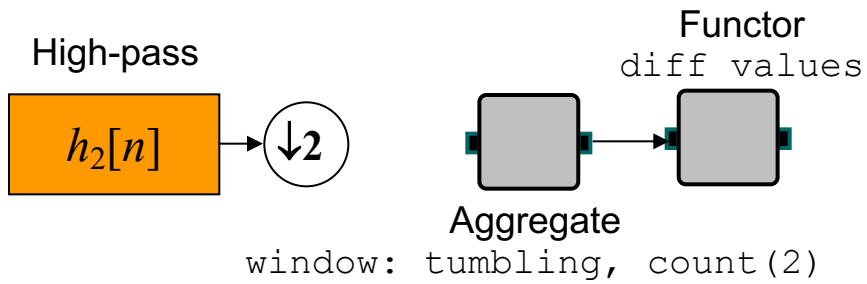
How does this relate to compressibility?

Implementing the Temporal Haar Transform

Applied window by window: For a given window, with a fixed size







Implementing the Temporal Haar Transform

- Fixed structure
  - if window size fixed, and known apriori can construct topology at compile time
- What can we do if window size unknown apriori?
  - Construct a transform matrix on the fly
  - Consider a 2 level Haar Transform. *Can we build a single transform matrix for this?*

$$\mathbf{U} = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

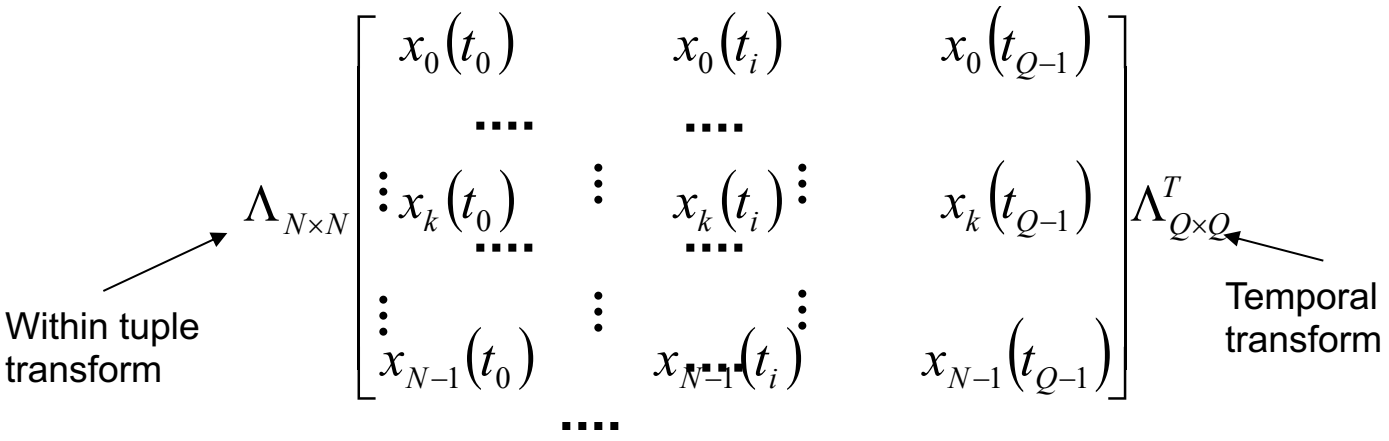
$$\Phi = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ \sqrt{2} & \sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & -\sqrt{2} \end{bmatrix}$$

Implementing the Haar Transform

- Temporal transform with unknown window size
  - Aggregate tuples for the window
  - Construct the transform matrix
  - Apply across the tuples (or tuple attributes)
- Within tuple transform
  - Can be applied tuple by tuple
  - Purely streaming – with transform matrix approach
  - Do not need any aggregation

From 1-D to 2-D Transforms

2-D Transform: Across tuples and across attributes within tuple



Collect tuples into one matrix $\mathbf{X}_{N\times Q}$

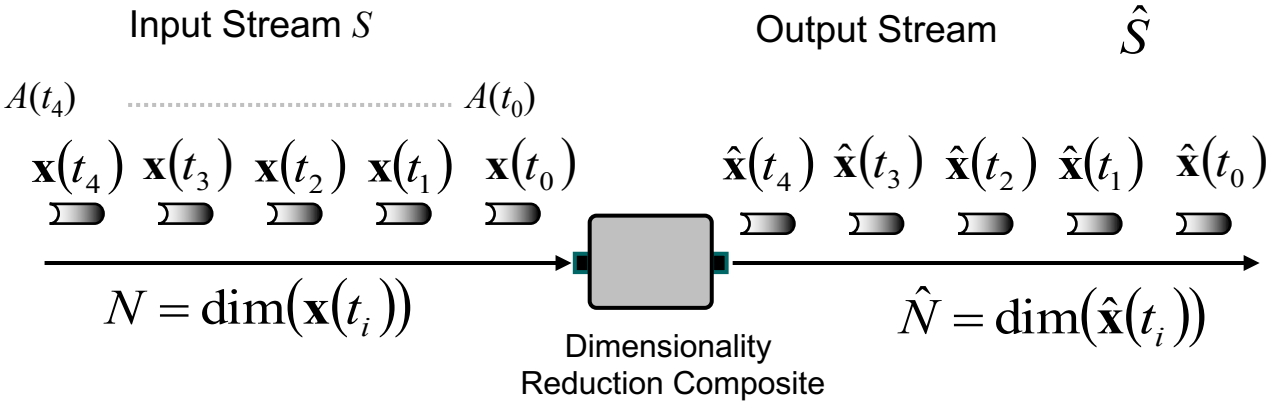
Assume Separability

Transforms: Summary

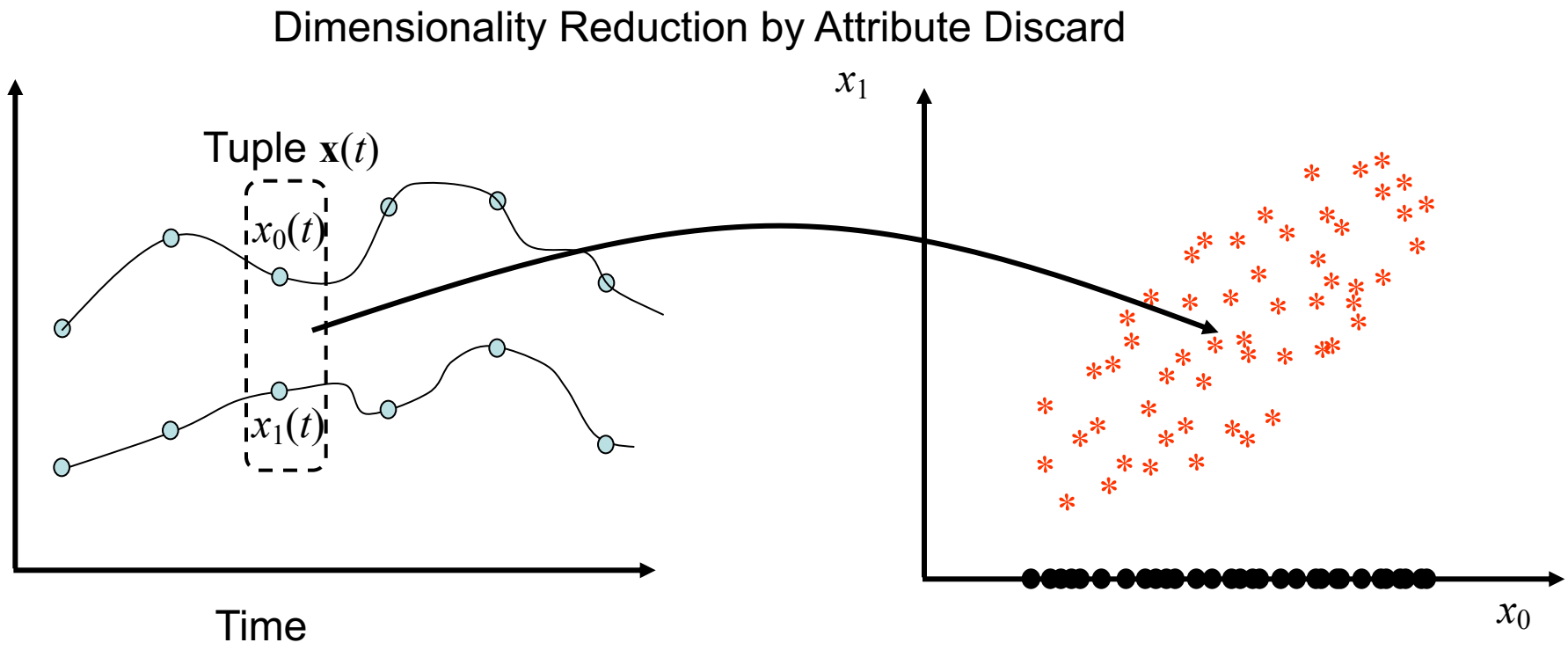
- Capture time-frequency properties of the signal
  - Allow different features to be extracted
- Allow approximating signal with smaller number of coefficients
  - Energy compaction
  - Link to dimensionality reduction and sketches
- Linear transforms
  - Provide windowed computation
  - Provide reversible representations

Dimensionality Reduction

- Applied within tuple
- Reduce number of attributes within tuple
  - From N to $\hat{N}$
- For numeric tuple
  - Appropriately transform the data into domain where attributes can be discarded
  - See link with energy compacting transforms

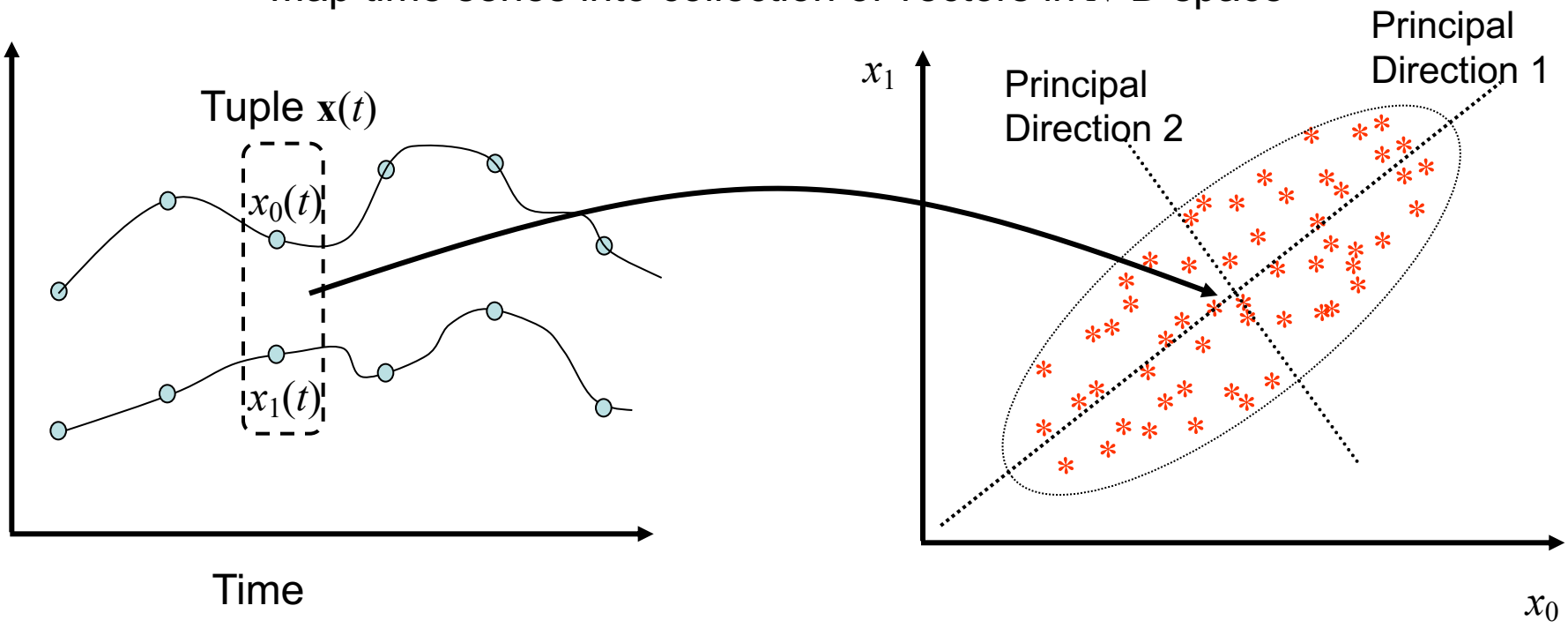


Dimensionality Reduction 2-D Example



Dimensionality Reduction: 2-D Example

Map time series into collection of vectors in N-D space



PCA identifies principal directions of variation in signal

Refresh: PCA and Karhunen-Loewe Transform

- Special kind of unitary transforms
  - Basis vectors are eigenvectors of covariance matrix
  - Decorrelates the dimensions
- Provides guaranteed smallest average squared reconstruction error
  - When only K ($<N$) coefficients are used to approximate signal
  - Consider that we look at Q time steps

$$\boldsymbol{\mu}_S = \frac{1}{Q}\sum_{j=0}^{Q-1}\mathbf{x}(t_j)$$

Sample Mean vector

$$\boldsymbol{\Sigma}_S = \frac{1}{Q}\sum_{j=0}^{Q-1}\left(\mathbf{x}(t_j)-\boldsymbol{\mu}\right)\left(\mathbf{x}(t_j)-\boldsymbol{\mu}\right)^T$$

Sample Covariance matrix

PCA and KLT

The basis vectors for KLT are eigenvectors of the covariance matrix of **s**

$$\boldsymbol{\Sigma}_S \mathbf{u}_k = \lambda_k \mathbf{u}_k$$

Using the relationship $\boldsymbol{\Sigma}_T = \mathbf{U}^H \boldsymbol{\Sigma}_S \mathbf{U}$ we have

$$\boldsymbol{\Sigma}_T = \begin{bmatrix} \mathbf{u}_0^H \\ \vdots \\ \mathbf{u}_{N-1}^H \end{bmatrix} \boldsymbol{\Sigma}_S \begin{bmatrix} \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathbf{u}_0^H \\ \vdots \\ \mathbf{u}_{N-1}^H \end{bmatrix} \begin{bmatrix} \lambda_0 \mathbf{u}_0 & \cdots & \lambda_{N-1} \mathbf{u}_{N-1} \end{bmatrix} = \begin{bmatrix} \lambda_0 & 0 & \cdots & 0 \\ 0 & \lambda_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_{N-1} \end{bmatrix}$$

Diagonal Matrix

Decorrelating transform

The eigenvalues are equal to the variance of transformed coefficients $\lambda_k = \sigma_{t_k}^2$

PCA and KLT

- Provides optimal squared error representation
  - Very efficient dimensionality reduction
  - Added de-correlating property
- Hard to compute in practice
  - Need all vectors available
  - Need to compute covariance matrix
  - Need to compute eigenvectors
  - Not directly suited for streaming data
- Incremental PCA algorithms
  - Approximate PCA

SPIRIT Algorithm

- SPIRIT: Streaming Pattern dIscoveRy in multIple Timeseries
- Transform based dimensionality reduction
  - Online algorithm: Compute the basis vectors (eigenvectors) incrementally instead of doing an eigen-decomposition
  - Can change amount of reduction incrementally
- Used in multiple applications
  - Forecasting
  - Pattern Detection
  - Anomaly Detection

SPIRIT Algorithm

Input dimensions N
Output dimensions $\hat{N}$

Initialization

Basis vectors

$$\mathbf{u}_k = \begin{bmatrix} \vdots \\ 1 \\ \vdots \end{bmatrix}$$

Energy (eigenvalue)

$$d_k = \varepsilon$$

$$0 \le k \le \hat{N} - 1$$

$$\mathbf{z} = \mathbf{x}(t)$$

New Tuple

for k=0: $\hat{N}$ -1

Iterate through the reduced number of dimensions

Iterative Update

$$\lambda_k = (\mathbf{z})^T \mathbf{u}_k$$

Project onto k-th basis vector

$$d_k = (\lambda_k)^2 + \alpha d_k$$

Update energy of k-th basis vector

SPIRIT Algorithm

$\mathbf{z} = \mathbf{x}(t)$ — New Tuple

for k=0: $\hat{N}$ -1 — Iterate through the reduced number of dimensions

Iterative Update

$\lambda_k = (\mathbf{z})^T \mathbf{u}_k$ — Project onto k-th basis vector

$d_k = (\lambda_k)^2 + \alpha d_k$ — Update energy of k-th basis vector

$\mathbf{e}_k = \mathbf{z} - \lambda_k \mathbf{u}_k$ — Find approximation error

$\mathbf{u}_k = \mathbf{u}_k + \frac{1}{d_k} \lambda_k \mathbf{e}_k$ — Update basis vector

$\mathbf{z} = \mathbf{z} - \lambda_k \mathbf{u}_k$ — Update basis vector

end

submit vector $\begin{bmatrix} \lambda_0 & \dots & \lambda_{\hat{N}-1} \end{bmatrix}$

Dimensionality Reduction

- SPIRIT
  - Can produce approximations to PCA
  - Number of vectors can be changed incrementally (based on error measures)
- Other techniques
  - MUSCLES
  - Distributed Dimensionality Reduction
  - SVD based dimensionality reduction

Summary

Techniques	Tuple Types	Windowing
Sketches	Numeric or non-numeric Tuples, Univariate or Multivariate	Not windowed. Can be tumbling window
Quantization	Numeric tuples. Univariate (scalar) or Multivariate (vector)	Tumbling window based
Transforms	Numeric tuples. Univariate (across tuples) or multivariate (within tuples) or both	Not windowed. Temporal transforms typically tumbling window
Dimensionality Reduction	Numeric tuples. Multivariate	Not windowed. Can be tumbling window

Wrapup

- Pre-processing Algorithms
  - Sketches
  - Quantization
  - Transforms
  - Dimensionality Reduction
- Next lecture
  - Stream mining and modeling algorithms
  - Classification
  - Regression

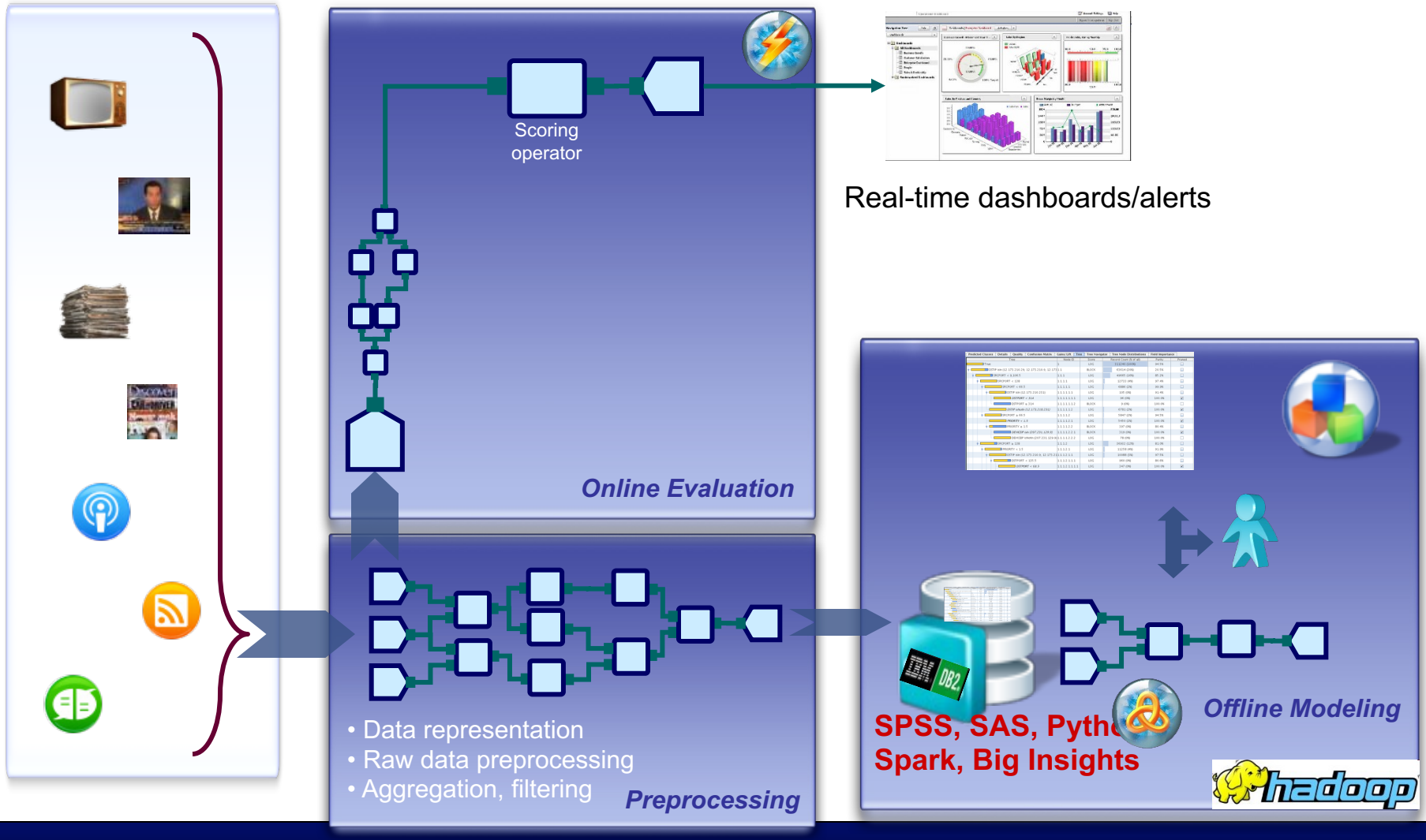
Modeling and Evaluation

- Modeling (model learning)
  - the process of applying knowledge discovery algorithms to the data to extract patterns and trends of interest, resulting in one or more *model*
  - Example: Clustering
    - data: grades of students
    - algorithm: *k*-means
    - model: k groups with mean grades
  - Usually the model is small compared to the data
- Two kinds
  - Supervised: training data is labeled (e.g., classification, regression)
  - Unsupervised: otherwise (e.g., clustering, frequent pattern mining)
- Various tools and libraries exist for modeling
  - Python, Spark, Matlab, R, SPSS, SAS, Weka

Modeling and Evaluation

- Model scoring (evaluation)
  - using the generated model on new data items, to predict what patterns they match
  - Example: Given a new grade, determine which category it belongs
- Model scoring is usually fast
- Model scoring is naturally a stream processing task

Offline Modeling and Online Evaluation



Offline Modeling and Online Evaluation

- Data Mining Group (DMG) has an XML-based vocabulary for defining models
  - PMML: Predictive model markup language
- Advantage:
  - Can use one system/tool to generate the model and another system to score it

Part of the PMML model

```
<MiningSchema>
  <MiningField usageType="active" name="AGE" importance="0.249"/>
  <MiningField usageType="active" name="NBR_YEARS_CLI" importance="0.065"/>
  <MiningField usageType="active" name="AVERAGE_BALANCE" importance="0.18"/>
  <MiningField usageType="active" name="MARITAL_STATUS" importance="0.024"/>
  <MiningField usageType="active" name="PROFESSION" importance="0.023"/>
  <MiningField usageType="active" name="BANKCARD" importance="0.449"/>
  <MiningField usageType="supplementary" name="GENDER"/>
  <MiningField usageType="predicted" name="ONLINE_ACCESS"/>
</MiningSchema>
```

Offline Modeling and Online Evaluation

Usage of the PMML model on Streams

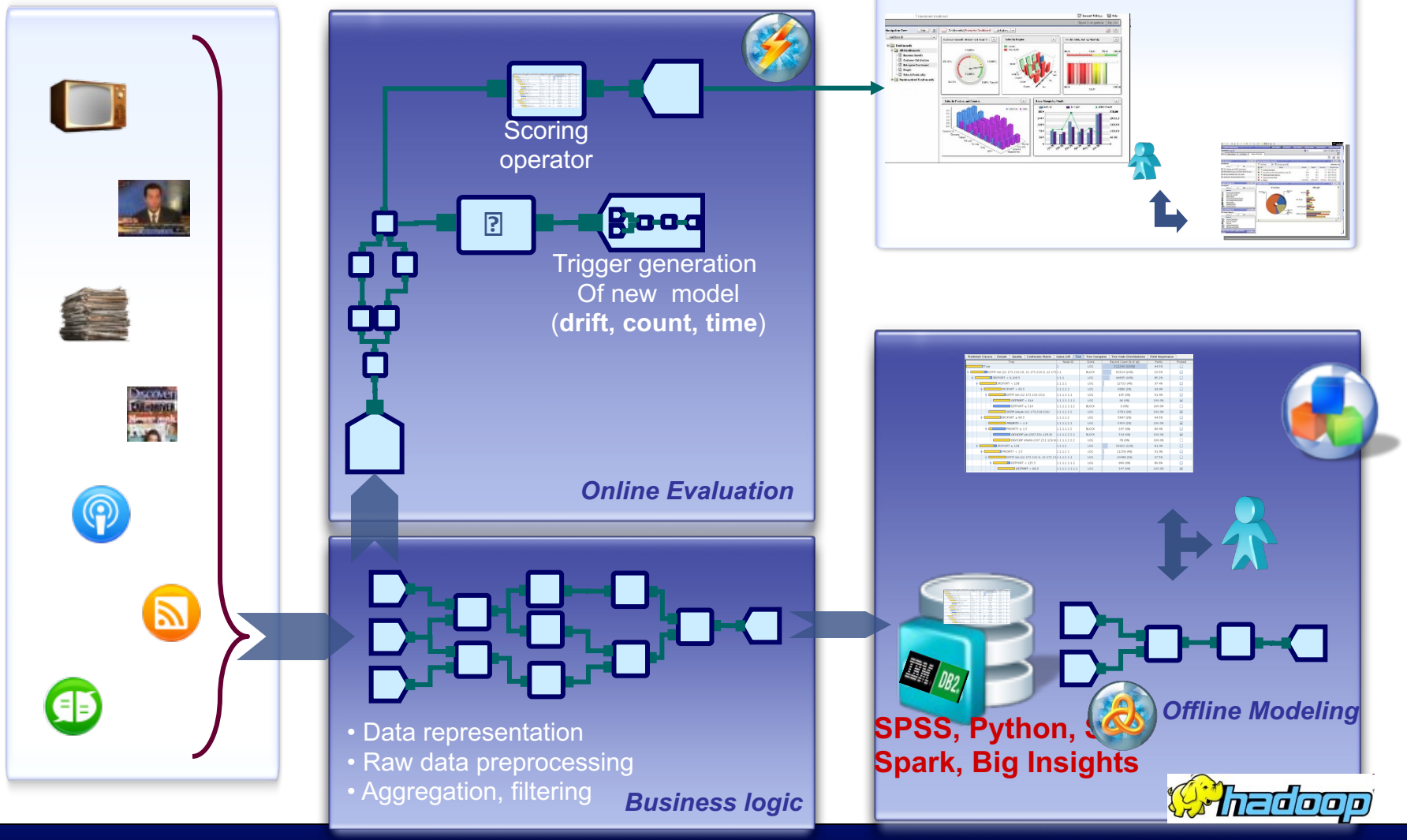
```
1   type InType = tuple<int32 age, int32 yearsCli, float64 avgBalance,
2      rstring maritalStatus, rstring profession, bool hasBankcard>;
3
4   stream<InType, tuple<rstring predictedClass, float64 confidence>> Out
5     = Classification(In)
6   {
7     param
8       model: "bankonlineaccess.pmml";
9       age: "AGE";
10      yearsCli: "NBR_YEARS_CLI";
11      avgBalance: "AVERAGE_BALANCE";
12      maritalStatus: "MARITAL_STATUS";
13      profession: "PROFESSION";
14      hasBankcard: "BANKCARD";
15  }
```

PMML model

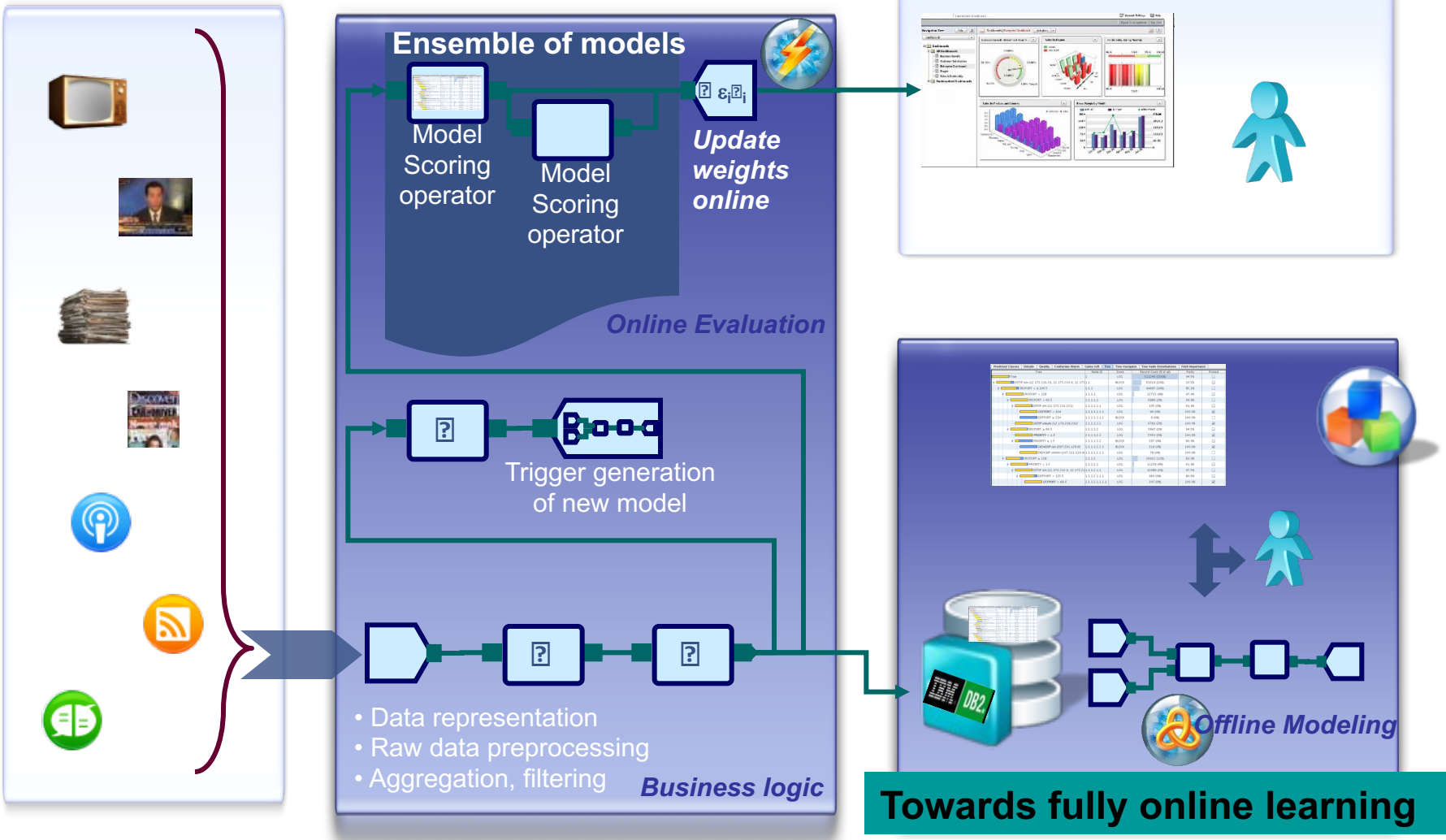
From Offline to Online Modeling

- Offline modeling and online scoring
  - Can be made adaptive by model re-learning and live model switch-over
  - What is the ideal period? Costly to adapt to short term changes.
- Online modeling and online scoring
  - The streaming way of doing things
  - Quick adaptation to changes in the patterns
  - Not always possible to come up with streaming algorithms
- Is there something in between?
  - Online Ensembles

From Offline to Online Analysis



From Offline to Online Analysis



Reading

- Book – Chapter 10: Preprocessing Algorithms
- Other Reading
  - Sampling and Summarization: Data Streams: Models and Algorithms (Chapter 2, Chapter 9, Chapter 12)
    - Sampling (Uniform, Non-uniform, threshold based)
      - Signal Processing Text
    - Reservoir Sampling
      - http://www.cs.umd.edu/~samir/498/vitter.pdf
    - Linear Transforms
      - Signal Processing Text (Alan V. Oppenheim, Ronald W. Schafer, John R. Buck : Discrete-Time Signal Processing)
  - Quantization: Signal Processing Text
    - Task specific Quantization for Speaker Verification
      - H. Tseng et al, “Quantization for Adapted GMM-based speaker verification”, ICASSP 2006.
    - Moment Preserving Quantization
      - E. Delp, M. Saenz, and P. Salama. "Block Truncation Coding (BTC)," The Handbook of Image and Video Processing. Academic Press, 2000.