Feature Engineering of Sensory Data

MLQS: Chapter 4

This slide-deck is a modified version of the authors' slides from the MLQS book

Overview

- Previously: data collection & pre-processing (e.g., removing noise)
- Today: extracting useful features
  - Time domain: Numerical, nominal, pattern mining
  - Frequency domain: Fourier analysis
  - Time + frequency: Wavelet analysis
  - Text data processing
  - Mobility data processing

Time point	Heart rate	Activity level	Speed	Activity type	Tired
0	45	low	0	inactive	no
1	120	high	10	running	no
2	45	low	0	inactive	no
3	120	high	10	running	no
4	120	high	9	running	yes
5	80	medium	5	walking	yes
6	45	low	0	inactive	no
7	80	medium	5	walking	no

Feature Engineering

Time point	Heart rate	Activity level	Speed	Activity type	Tired
0	45	low	0	inactive	no
1	120	high	10	running	no
2	45	low	0	inactive	no
3	120	high	10	running	no
4	120	high	9	running	yes
5	80	medium	5	walking	yes
6	45	low	0	inactive	no
7	80	medium	5	walking	no

Feature is an individual measurable property or characteristic of a phenomenon being observed. Choosing informative, discriminating and independent features is a crucial step for effective algorithms

https://en.wikipedia.org/wiki/Feature_(machine_learning)

Time Domain

- Imagine the following sequence

Time point	Heart rate	Activity level	Speed	Activity type	Tired
0	45	low	0	inactive	no
1	120	high	10	running	no
2	45	low	0	inactive	no
3	120	high	10	running	no
4	120	high	9	running	yes
5	80	medium	5	walking	yes
6	45	low	0	inactive	no
7	80	medium	5	walking	no

Types of data

Numerical data or quantitative data

= numerical values

Discrete data

If you can count it, then it is discrete:

Continuous data

If you can measure it, then it is continuous:

* length
* weight
* temperature

etc

Categorical data or qualitative data

= categories

Nominal data

If you can brand it, then it is nominal:

Gender	Colour
Female	Blue
Male	Red
	Green
	Orange

Ordinal data

If you can order or rank it, then it is ordinal:

- Always
- Usually
- Sometimes
- Rarely
- Never

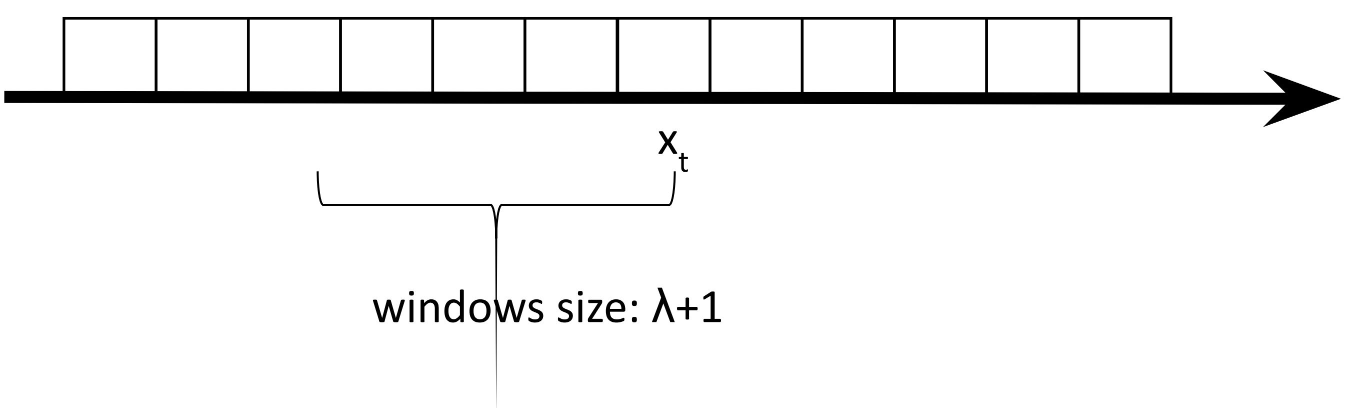
(Likert scales)

https://legac.com.au/blogs/further-mathematics-exam-revision/further-mathematics-unit-3-data-analysis-types-of-data

Time Domain: numerical (1)

- Summarize values of a numerical attribute *i* in a given window
- Assume a temporal ordering in the dataset: $x_1^i, \ldots, x_N^i$
- Need to select **a windows size parameter λ** (i.e., # instances or samples per window = λ+1)
- For each time point *t,* extract features, $x\_new^i_t$: using the windowed dataset:

$$[x_{t-\lambda}^i, \ldots, x_t^i$$



Time Domain: numerical (2)

- Compute a new value per time point of a feature over each of these values (sliding window):

$$x\_mean_t^i = \frac{\sum_{n=t-\lambda}^{t} x_n^i}{\lambda + 1}$$

$$x\_max_t^i = max_{t-\lambda \leq n \leq t} x_n^i$$

$$x\_min_t^i = min_{t-\lambda \leq n \leq t} x_n^i$$

$$x\_std_t^i = \sqrt{\frac{\sum_{n=t-\lambda}^{t} (x\_mean_n^i - x_t^i)^2}{\lambda + 1}}$$

Time Domain: numerical (3)

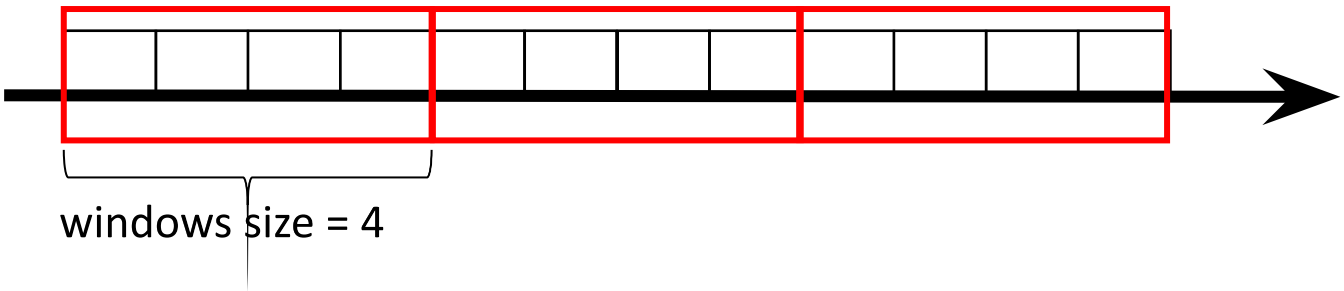
- Example outcome (w/ window parameter λ = 1)

$$[x^i_{t-\lambda}, \ldots, x^i_t$$

Time point	Heart rate	Temporal mean heart rate	Tired
0	45	-	no
1	120	82.5	no
2	45	82.5	no
3	120	82.5	no
4	120	120	yes
5	80	100	yes
6	45	62.5	no
7	80	62.5	no

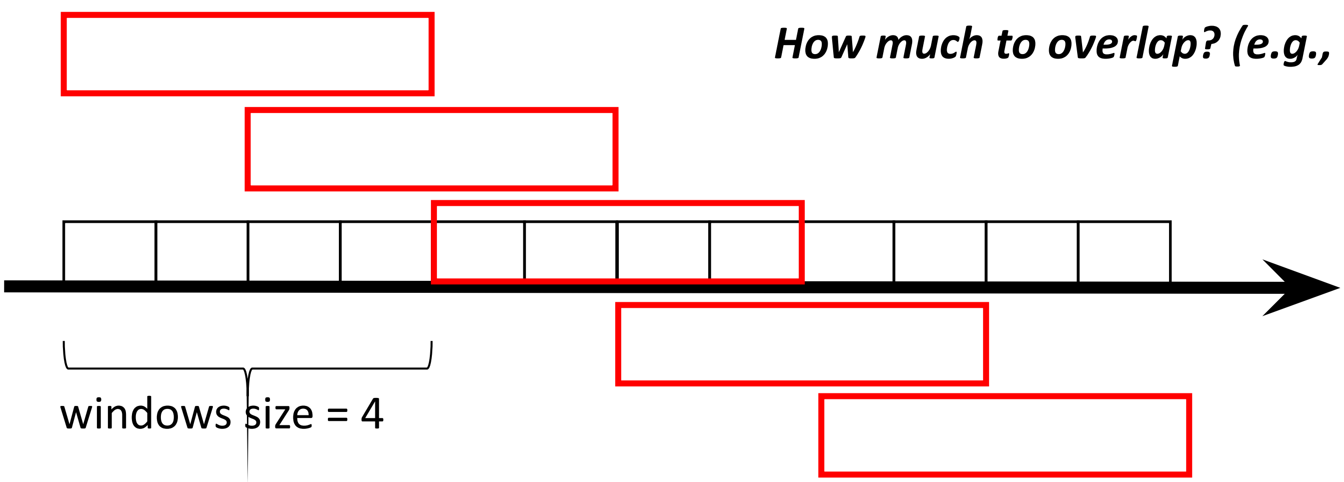
Distinct vs. Overlapped Windowing

- Distinct windows



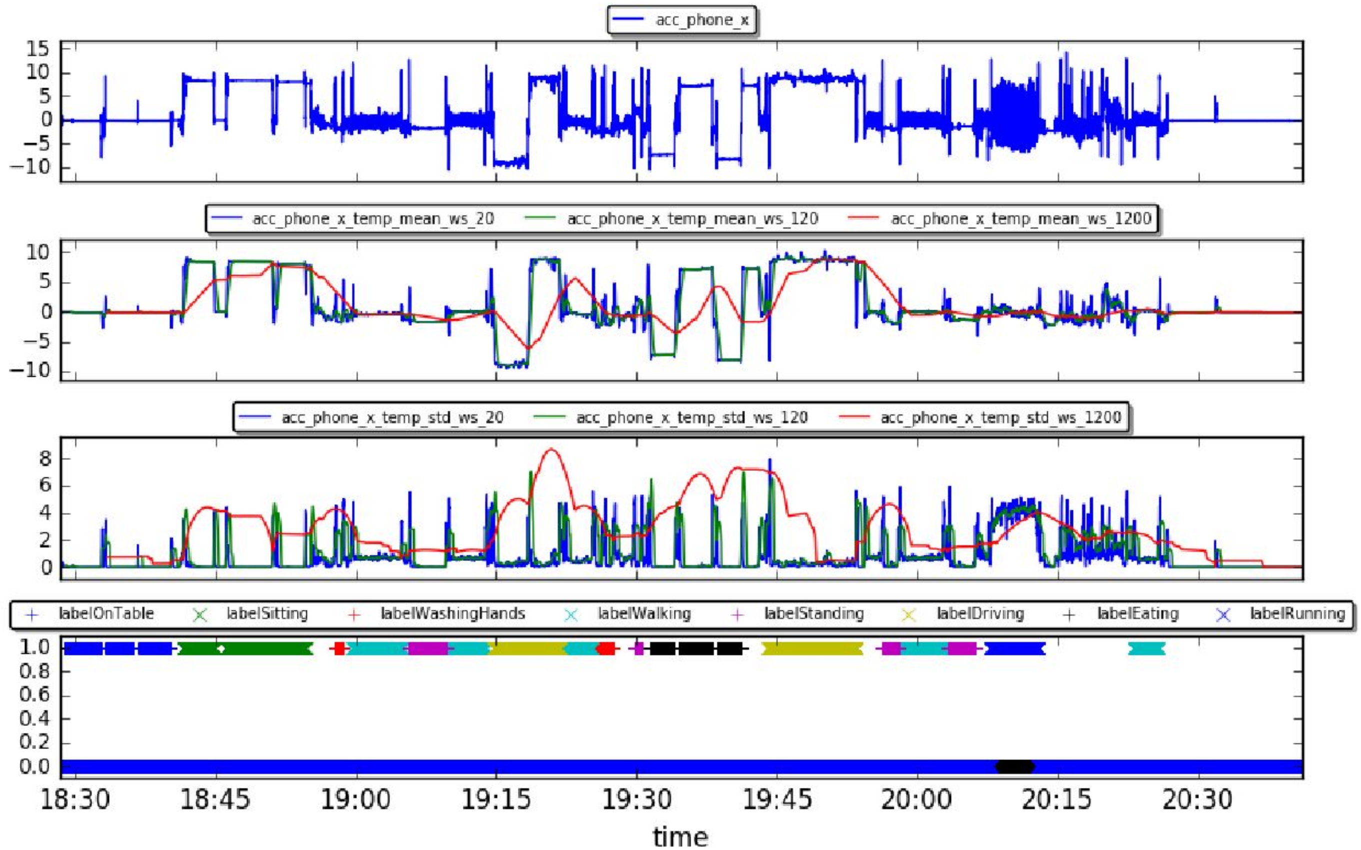
- Overlapping windows

How much to overlap? (e.g., 50%)

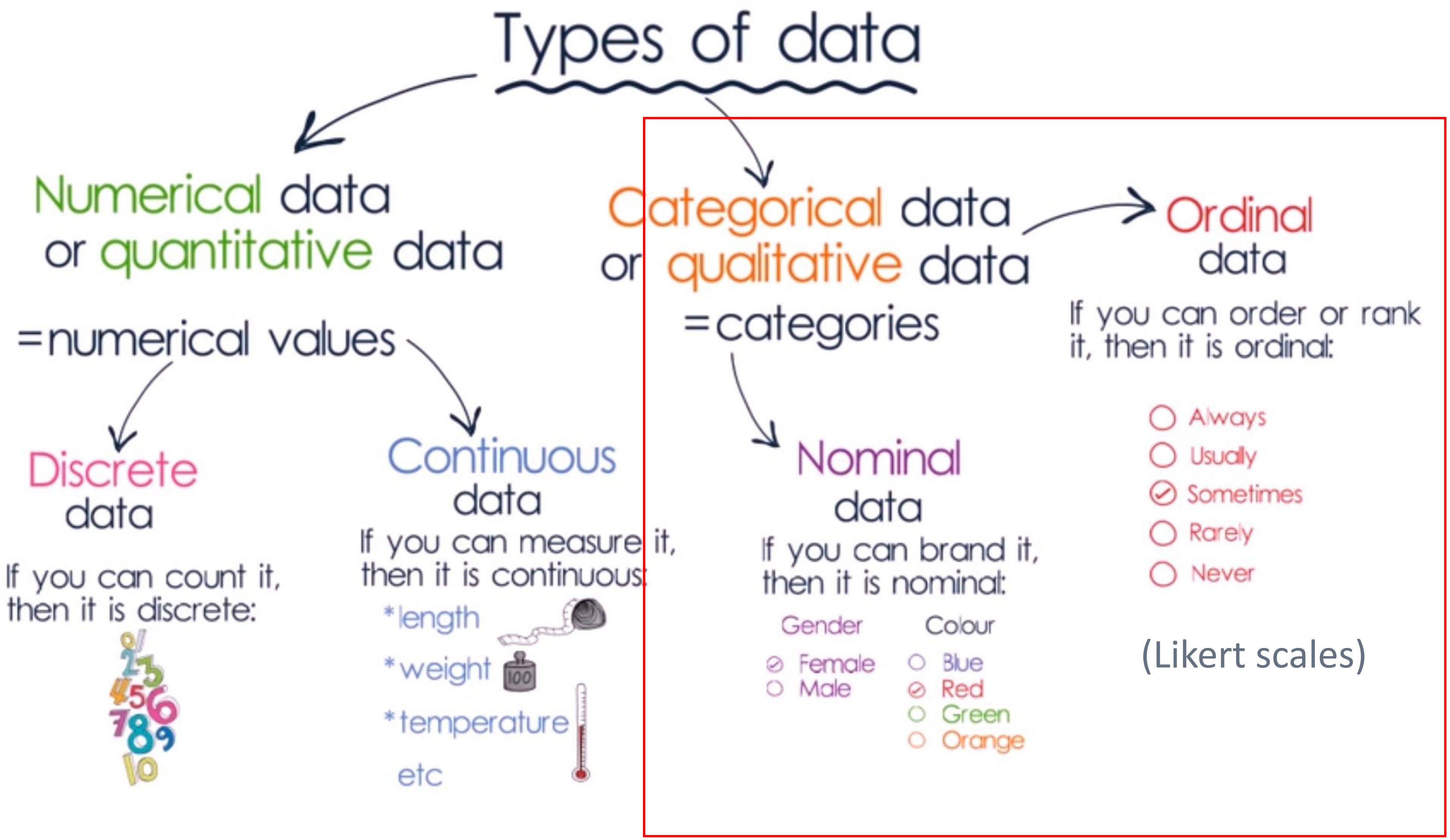


Time Domain: numerical (4)

- CrowdSignals data:
  - Numerical temporal aggregation with different window sizes
    (a window size of 20 resembles 5 s, 120 is 30 s, and 1200 is 5min)



Types of data



https://legac.com.au/blogs/further-mathematics-exam-revision/further-mathematics-unit-3-data-analysis-types-of-data

Time Domain: categorical / nominal (1)

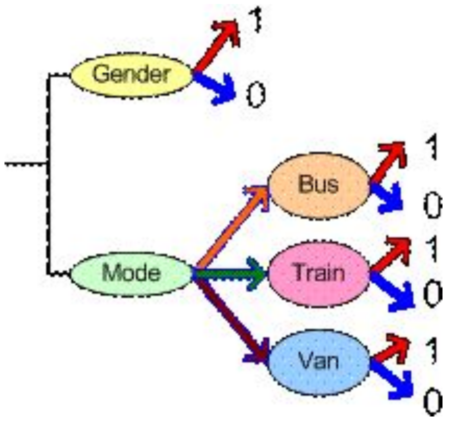
- What if we have categorical data?
  - Screen on/off events
  - Phone charging on/off events
  - Types of apps used
  - Types of activities performed
  - Types of current locations (e.g., home, work)
- These categorical data (likely to have some semantic meaning) are typically from:
  - events (e.g., app usage) or
  - semantic data generated from low-level sensor data (e.g., GPS clustering – significant place, activity classification– running, inactive)

Time Domain: categorical / nominal (2)

- Let's consider screen on/off events
- From screen on/off events for a given window, we can find the following features:
  - (numeric) Duration that a certain state lasts
  - (numeric) Frequency that a certain state occurs or is switched to others
  - (nominal) State at a given point
- Numeric values can be readily used for machine learning, but "**nominal**" values cannot be used directly

Time Domain: categorical / nominal (3)

- How can use “nominal” values as features?
  - *Answer: by doing OneHotEncoding* of nominal data that represents each name w/ a dummy variable
  - Dummy variable is a variable that can assume either one of two values (usually 1 and 0), where 1 represents the existence of a certain condition and 0 indicates that the condition does not hold



- Example: Yes, No, NA
  - *Yes*: *VAL_YES = 1, VAL_NO = 0, VAL_NA = 0*
  - *No*: *VAL_YES = 0, VAL_NO = 1, VAL_NA = 0*
  - *NA*: *VAL_YES = 0, VAL_NO = 0, VAL_NA = 1*

Time Domain: categorical / nominal (3)

- Given that we have k categories (values), one-hot encoding allows for k degrees of freedom, while the variable itself needs only k–1
- Dummy coding removes the extra degree of freedom by using only k–1 features in the representation

Table 5-3. Toy dataset of apartment prices in three cities

	City	Rent
0	SF	3999
1	SF	4000
2	SF	4001
3	NYC	3499
4	NYC	3500
5	NYC	3501
6	Seattle	2499
7	Seattle	2500
8	Seattle	2501

Feature Engineering for Machine Learning_ Principles and Techniques for Data Scientists

Time Domain: categorical / nominal (3)

- One-hot encoding allows for k degrees of freedom, while the variable itself needs only k–1. Dummy coding removes the extra degree of freedom by using only k–1 features in the representation

Table 5-1. One-hot encoding of a category of three cities

	e_1	e_2	e_3
New York	1	0	0
San Francisco	0	1	0
Seattle	0	0	1

```
one_hot_df = pd.get_dummies(df, prefix=['city'])
```

Table 5-2. Dummy coding of a category of three cities

	e_1 (SF)	e_2 (Seattle)
New York	0	0
San Francisco	1	0
Seattle	0	1

```
dummy_df = pd.get_dummies(df, prefix=['city'],
drop_first=True)
```

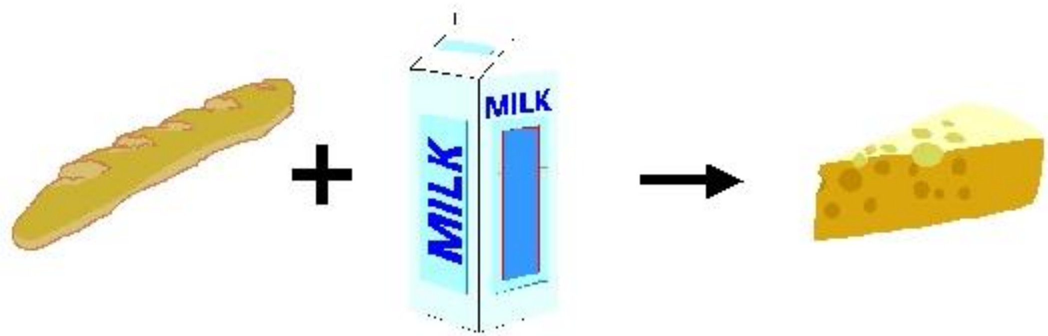
Time Domain: categorical / pattern mining

- What if we have categorical data?
- Generate temporal patterns that combine categorical values over time
- Follow an approach by Batal *et al.* (2013)
- Again consider a window size λ
- Consider different temporal patterns:
    - succession (denoted as b)
    - co-occurrence (denoted as c)

Batal et al., A Temporal Pattern Mining Approach for Classifying Electronic Health Record Data, ACM Trans Intell Syst Technol. 2013 Sep; 4(4):

Association-Rule Mining

- Flagship of data mining
- What items are frequently bought together by customers?

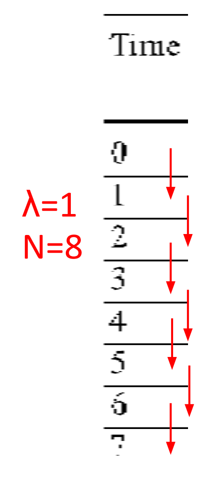


http://infolab.stanford.edu/~evtimov/Defense/sld009.htm

Support (1)

- Example patterns:
  - Activity level = low [**c**: co-occurs w/] Activity = running
  - Activity = running [**b**: succession] Activity = running
- How do we find these patterns?
  - Consider the notion of support
  - What fraction of all time points does the pattern occur?

$$support(pa) = \frac{\sum_{t=t_{start}+\lambda}^{t_{end}} occurs(pa, t-\lambda, t)}{N-\lambda}$$



Support (2)

- Support (w/ λ=1) for
  "*Activity level* = *low* **[b: succession]** *Activity type* = *running*"

Two samples per window

Time point	Heart rate	Activity level	Speed	Activity type	Tired
0	45	low	0	inactive	no
1	120	high	10	running	no
2	45	low	0	inactive	no
3	120	high	10	running	no
4	120	high	9	running	yes
5	80	medium	5	walking	yes
6	45	low	0	inactive	no
7	80	medium	5	walking	no

Support = 2/7 ≈ 0.29

Support (3)

- How to formally define whether a pattern occurs?

$$
occurs(pa, t_s, t_e) = \begin{cases} 1 & \begin{array}{r} \text{(1) } pa \text{ of the form } X_i = v \text{ and there exists a time point} \\ \text{between } t_s \text{ and } t_e \text{ where } v \text{ is observed for } X_i \\ \text{(2) } pa \text{ is of the form } pa_1 \text{ (c) } pa_2 \text{ and there exists a time point} \\ \text{between } t_s \text{ and } t_e \text{ where both } pa_1 \text{ and } pa_2 \text{ occur} \\ \text{(3) } pa \text{ is of the form } pa_1 \text{ (b) } pa_2 \text{ and there exists a time point } t_1 \\ \text{before } t_2 \text{ both between } t_s \text{ and } t_e \text{ such that } pa_1 \text{ occurs at } t_1 \\ \text{and } pa_2 \text{ at } t_2 \end{array} \\ 0 & \text{otherwise} \end{cases}
$$

Pattern generation (1)

- Focus only on patterns with sufficient support
  - Otherwise it will not be a good feature anyway
  - Start with patterns of single attribute value pairs with sufficient support
  - Extend these patterns to more complex patterns and select those with sufficient support
  - As we move to more **complex patterns of size *k*,** we only extend patterns of size *k-1* that were among the ones with sufficient support

Pattern generation (2)

Algorithm 1: Temporal Pattern Identification Algorithm

```
P = {}
k = 1
Generate patterns of size 1 (attribute values pairs)
Calculate the support for each pattern and add the ones that reach the threshold θ to P
while True do
    Select the current set of k-patterns P_k from P
    Try to extend each element of P_k with an element from P_1 using (c) and (b) constructs
    Calculate the support for the new cases
    Add the cases to the set P for which the support ≥ θ
    k = k + 1
    if no cases have been added then
        return P
    end
end
```

Pattern generation (3)

- Let us consider our dataset again

Time point	Heart rate	Activity level	Speed	Activity type	Tired
0	45	low	0	inactive	no
1	120	high	10	running	no
2	45	low	0	inactive	no
3	120	high	10	running	no
4	120	high	9	running	yes
5	80	medium	5	walking	yes
6	45	low	0	inactive	no
7	80	medium	5	walking	no

- Assume a minimum support threshold $\Theta=2/7$ and window size parameter of $\lambda=1$ (i.e., two samples per window)
- What patterns would we get if we only consider **Activity type**?

Pattern generation (4)

{inactive, inactive}
{inactive, walking}
{inactive, running}

{walking, walking}
{walking, inactive}
{walking, running}

{running, running}
{running, inactive}
{running, walking}

- The resulting dataset:

k=1 **k=2**

Time point	Heart rate	Activity level	Speed	Activity type	*Activity type = inactive*	*Activity type = running*	*Activity type = walking*	*Activity type = inactive (b) Activity type = running*	Tired
0	45	low	0	inactive	-	-	-	-	no
1	120	high	10	running	1	1	0	1	no
2	45	low	0	inactive	1	1	0	0	no
3	120	high	10	running	1	1	0	1	no
4	120	high	9	running	0	1	0	0	yes
5	80	medium	5	walking	0	1	1	0	yes
6	45	low	0	inactive	1	0	1	0	no
7	80	medium	5	walking	1	0	1	0	no
					5/7	**5/7**	**3/7**	**2/7**	

$$support(pa) = \frac{\sum_{t=t_{start}+\lambda}^{t_{end}} occurs(pa, t-\lambda, t)}{N-\lambda}$$

Time Domain: categorical / pattern mining

- CrowdSignals data (labels, λ=1200; 5min, Θ=0.03)

1-patterns (7)	2-patterns (10)
OnTable, Sitting, Walking, Standing, Driving, Eating, Running	OnTable (b) OnTable, Sitting (b) Sitting, Walking (b) Walking, Walking (b) Standing, Walking (b) Driving, Standing (b) Walking, Standing (b) Standing, Driving (b) Driving, Eating (b) Eating, Running (b) Running

Time Domain: mixed data

- We can make categories from the numerical values
  - Use ranges (low, normal, high)
  - Use temporal relations (increasing, decreasing)
- We apply the categorical approach to those

Overview

- Previously: data collection & pre-processing (e.g., removing noise)
- Today: creating useful features
  - Time domain: Numerical, nominal, pattern mining
  - Frequency domain: Fourier analysis
  - Time + frequency: Wavelet analysis
  - Text data processing
  - Mobility data processing

Frequency domain

- Next to summarizing the values we can also look at the frequency domain
  - Periodic data, e.g. a walking pattern
- Let us consider our series of values again within a certain window of size λ (# samples):
  $[x^i_{t-\lambda}, \ldots, x^i_t]$
- Perform a Fourier transformation to see what "frequencies" we observe within the window

Fourier transformation

- Decomposing signals w/ sinusoid functions
  - X(k): similarity between k-th sinusoidal basis functions and the original time series
    - *k* runs from 0 to the window size ($N = \lambda$)
    - Finding X(k) w/ fast Fourier transform (FFT)
  - Sampling frequency: f_s (how many samples per second)
  - X(k) corresponds to frequency **F(k) = f_s * k / N**

$$X(k) = \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}$$

$$X(k) = \sum_{n=0}^{N-1} x(n)\left(\cos\left(2\pi kn/N\right) - i\sin\left(2\pi kn/N\right)\right)$$

$$e^{-i\theta} = \cos\theta - i\sin\theta$$

- *Euler's formula*
  - *cos (2π*k*f_0*t) ⇔ cos (2π*k*n/N)*
  - *Here, f_0 is base frequency*
    - ⇒ *$f_0 t = n/\lambda$ – nth index*
- *$f_s = T / N$*
  - *T = time duration of a window*
  - *N = # samples per window*

And now…. get feature values

- The highest amplitude frequency:

$$x\_max\_f_t^i = f\left(\underset{k\in[0,\lambda]}{\operatorname{argmax}}\ X_{t\text{-}\lambda}(k)\ \right)$$

- Frequency weighted signal average:

$$x\_f\_weighted_t^i = \frac{\sum_{k=0}^{\lambda-1} X_{t\text{-}\lambda}(k) \cdot f(k)}{\sum_{k=0}^{\lambda-1} X_{t\text{-}\lambda}(k)}$$

- And the amount of information in the signal (power spectrum entropy):

$$P_{t-\lambda}^t(k) = \frac{1}{\lambda}\left|\mathrm{X}_{\mathrm{t\text{-}\lambda}}(\mathrm{k})\right|^2$$

$$p_{t-\lambda}^t(k) = \frac{P_{t-\lambda}^t(k)}{\sum_{i=0}^{\lambda-1} P_{t-\lambda}^t(i)}$$

$$x\_power\_spec\_entropy_t^i = -\sum_{k=0}^{\lambda} p_{t-\lambda}^t(k)\ ln\ p_{t-\lambda}^t(k)$$

Entropy

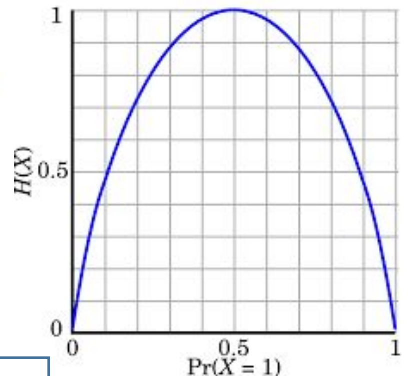
Suppose X can have one of *m* values… $V_1, V_2, \ldots V_m$

$P(X=V_1) = p_1$	$P(X=V_2) = p_2$	….	$P(X=V_m) = p_m$

What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from X's distribution? It's

$$H(X) = -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \ldots - p_m \log_2 p_m$$

$$= -\sum_{j=1}^{m} p_j \log_2 p_j$$



A histogram of the frequency distribution of values of X would be flat

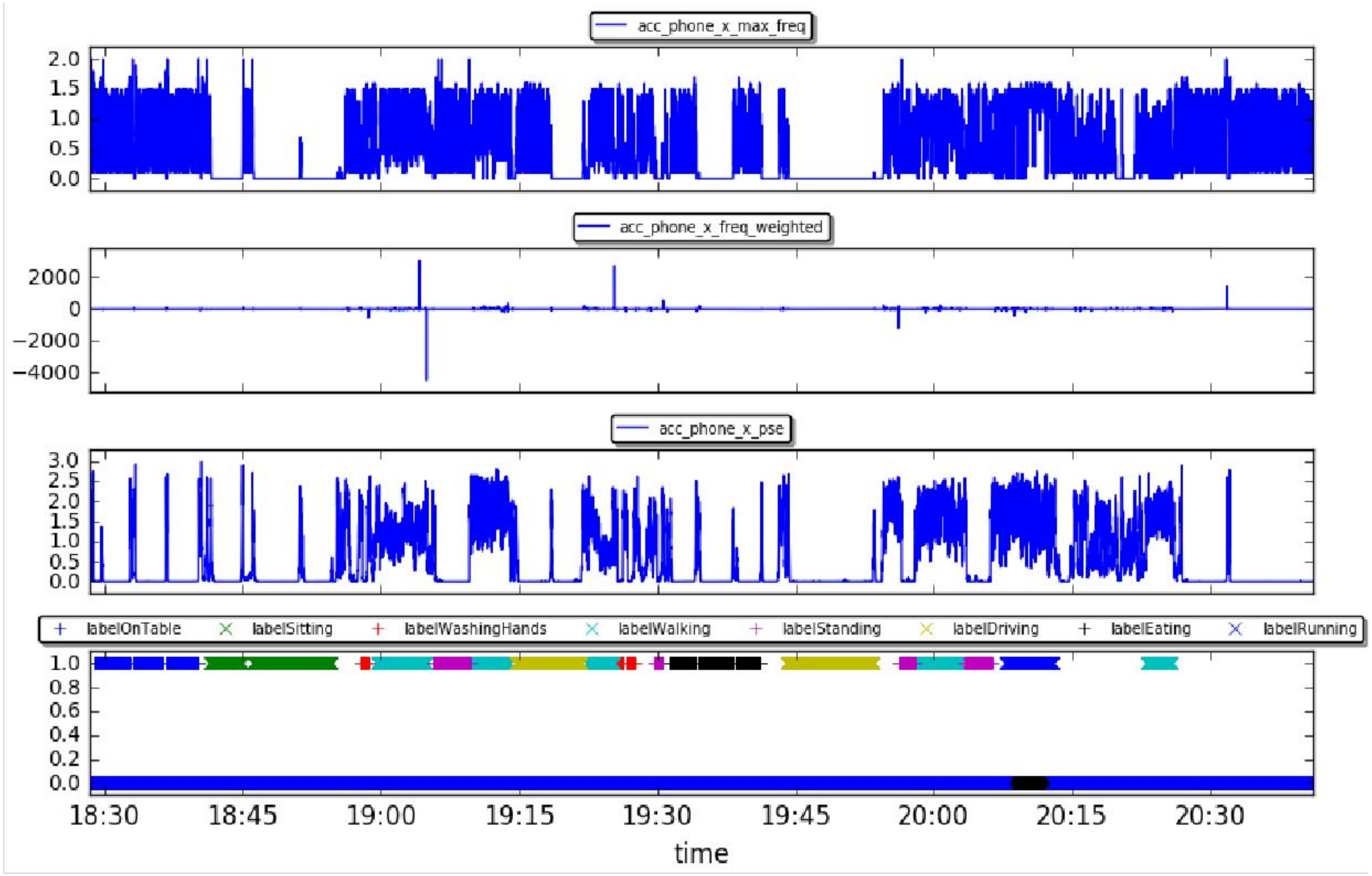
H(X) = The entropy of X

- "High Entropy" means X is from a uniform (boring) distribution
- "Low Entropy" means X is from varied (peaks and valleys) distribution

http://www.cs.cmu.edu/~./awm/tutorials/infogain.html

Frequency domain (2)

- CrowdSignal example (λ=40)

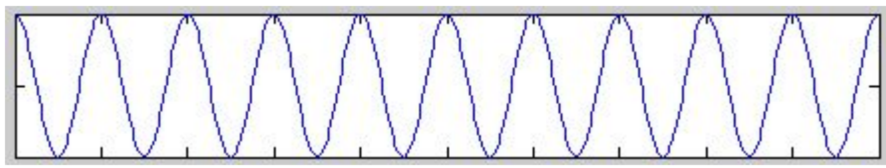


Overview

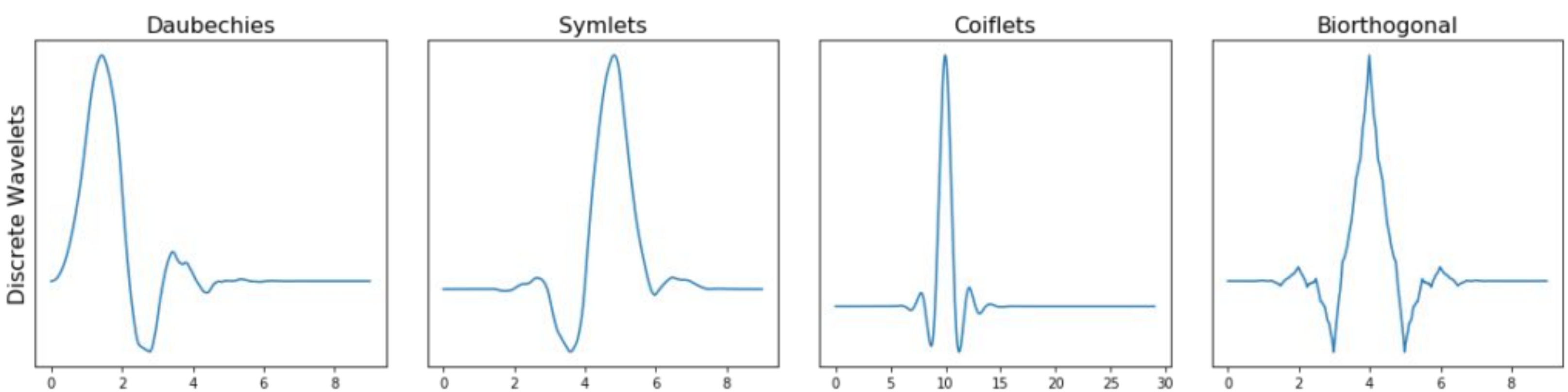
- Previously: data collection & pre-processing (e.g., removing noise)
- Today: creating useful features
  - Time domain: Numerical, nominal, pattern mining
  - Frequency domain: Fourier analysis
  - Time + frequency: Wavelet decomposition
  - Mobility data processing
  - Text data processing

Wavelet analysis

Fourier Analysis is based on sine/cosine basis functions



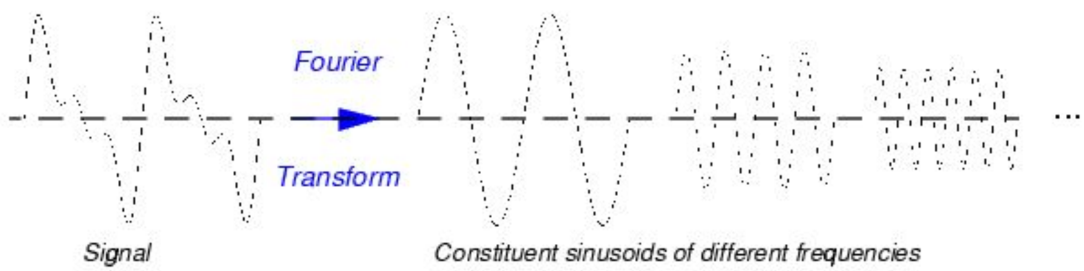
Wavelet analysis is based on more complex basis functions (called “wavelets)



Wavelets are generated from the single mother wavelet Ψ(t) by scaling s and shifting

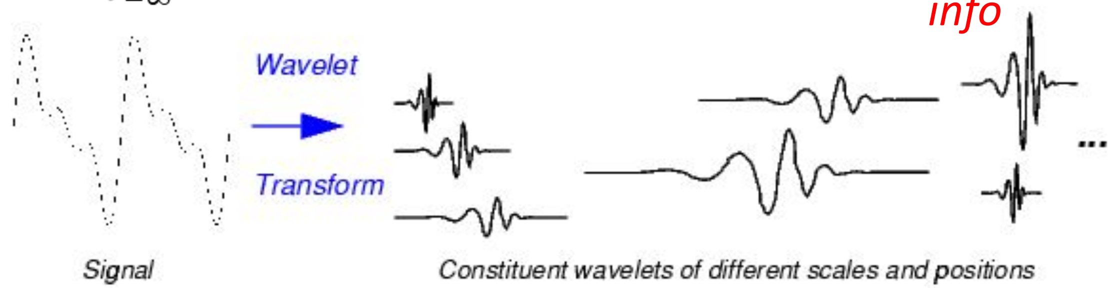
Wavelet vs. Fourier transform

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{j2\pi ux} du$$



$$f(t) = \int_{-\infty}^{\infty} C(scale, position) \psi(scale, position, t) dt$$

□ frequency + time info



- Wavelet transformation: spectral (‘frequency’) information and partly the information about the event in time (spatial coordinated in 2D)
- Fourier transformation: spectral (frequency) information only

https://en.wikipedia.org/wiki/Discrete_wavelet_transform
Tutorial: http://disp.ee.ntu.edu.tw/tutorial/WaveletTutorial.pdf

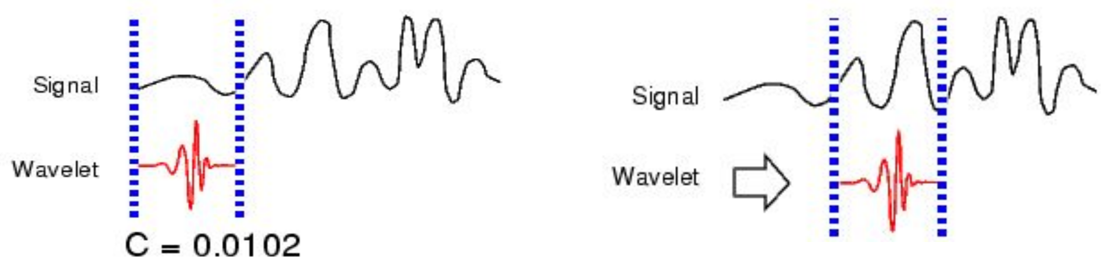
Wavelet transform

Scaling (frequency)

Rapid change
High frequency

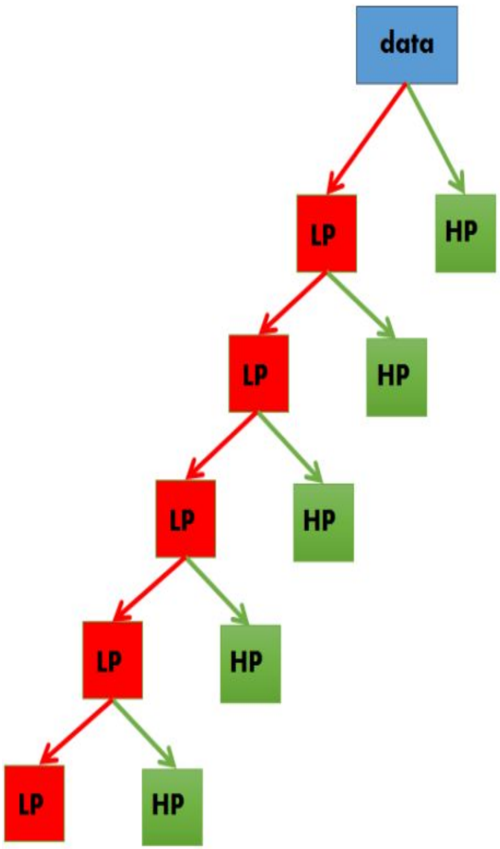
Slow change
Low frequency

Shifting (time)

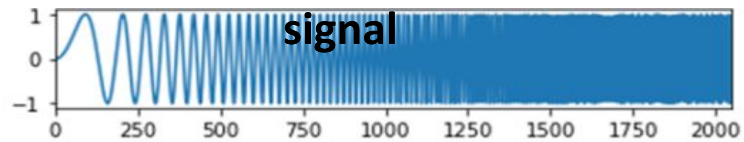


https://en.wikipedia.org/wiki/Discrete_wavelet_transform
Tutorial: http://disp.ee.ntu.edu.tw/tutorial/WaveletTutorial.pdf

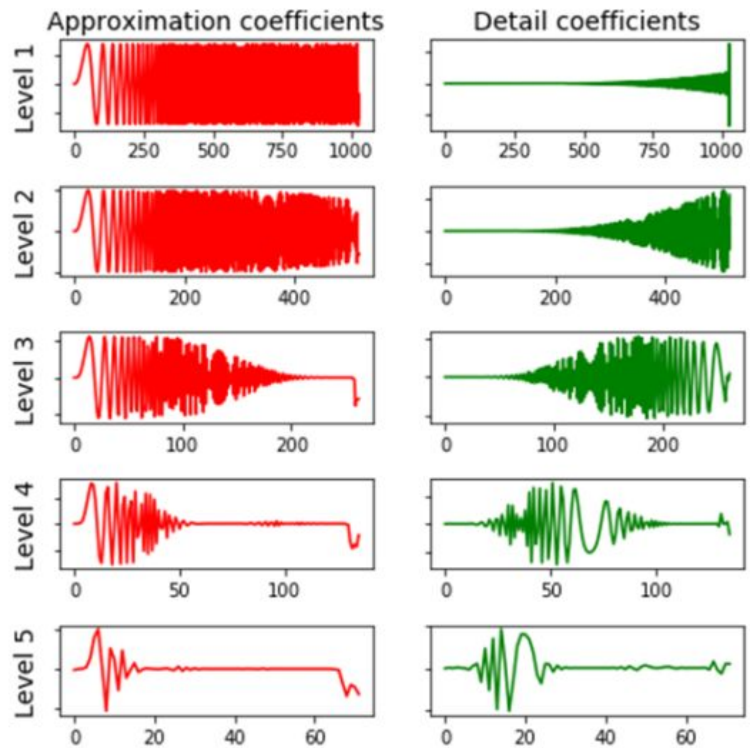
Filter bank approach for signal decomposition



Original signal



pyWavelets: pywt.dwt() w/ 5 levels



For each decomposed coefficient set, perform feature extraction (e.g., mean, standard deviation)

- http://ataspinar.com/2018/12/21/a-guide-for-using-the-wavelet-transform-in-machine-learning/
- TrailSense: A Crowdsensing System for Detecting Risky Mountain Trail Segments with Walking Pattern Analysis

Overview

- Previously: data collection & pre-processing (e.g., removing noise)
- Today: creating useful features
  - Time domain: Numerical, nominal, pattern mining
  - Frequency domain: Fourier analysis
  - Time + frequency: Wavelet analysis
  - Mobility data processing
  - Text data processing

Mobility data processing

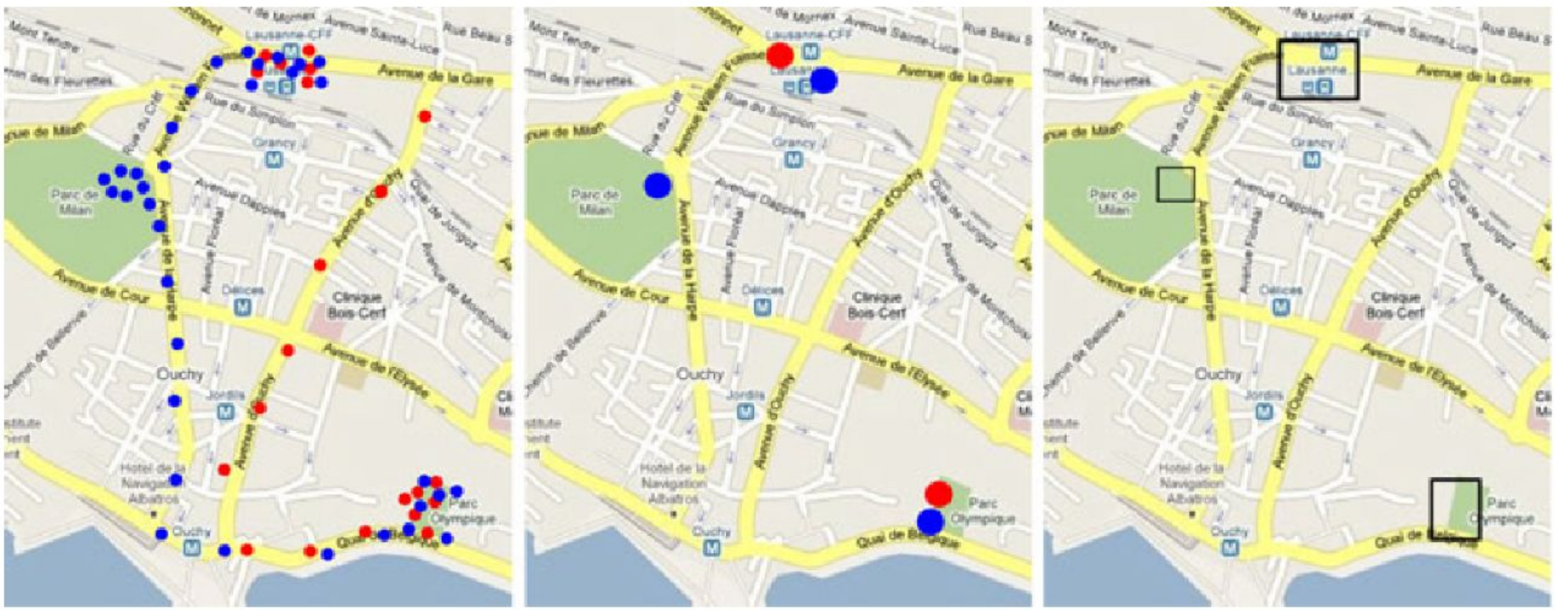
- Mobility data: GPS coordinates (i.e., latitude and longitude) and collection time
- The latitude and longitude seems like numerical values, but how can we use them for machine learning?
- One approach: finding semantically meaningful places and use this information for feature generation
  - How long did a user stay at home, or work?
  - How frequently did a user change places for the last three hours?
- How? Clustering GPS coordinates

Mobility data processing

- The purpose of clustering GPS coordinates to find informative locations, where people often visit or stay
- Clustering:
  - First need to define a **distance metric** that measures distance between two data points
    - Euclidean distance is widely-used; but, for GPS coordinates, we should use **Haversine distance (over ball shape)**
  - Well-known clustering algorithms (e.g., DBSCAN) can be used for clustering
- After completion of clustering, we can get cluster labels for each data point
- We then can extract features in a same manner as feature extraction on categorical values

https://en.wikipedia.org/wiki/Great-circle_distance

Mobility data processing



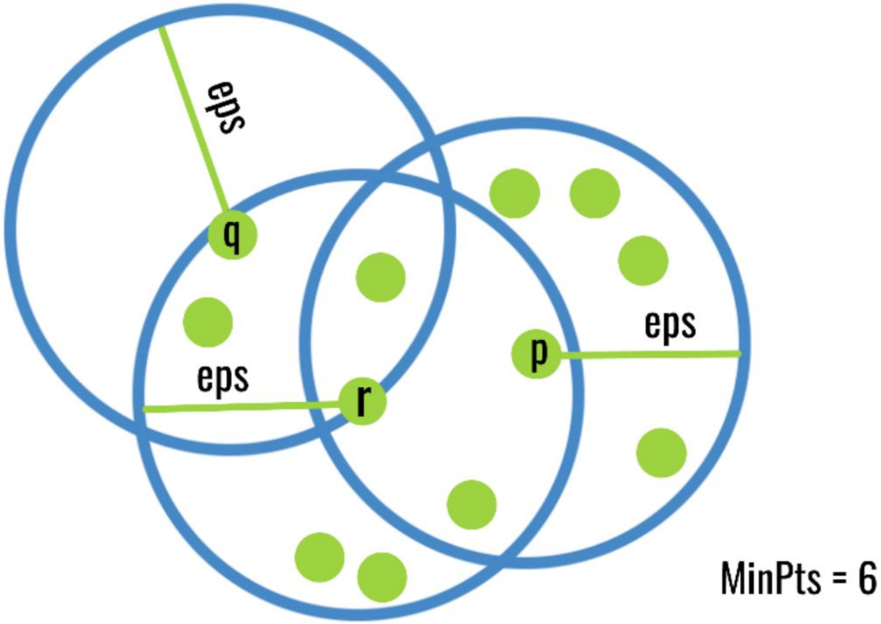
(a) Location Points (b) Stay Points (c) Stay Regions

Fig. 1 *Left*: location points obtained for a hypothetic user during two days (*red and blue*). *Middle*: stay points discovered for the two days. *Right*: stay regions estimated using the previous stay points as input data

Discovering places of interest in everyday life from smartphone data, Raul Montoliu, Jan Blom, Daniel Gatica-Perez, Multimed Tools Appl (2013) 62:179–207

Density-based Clutering

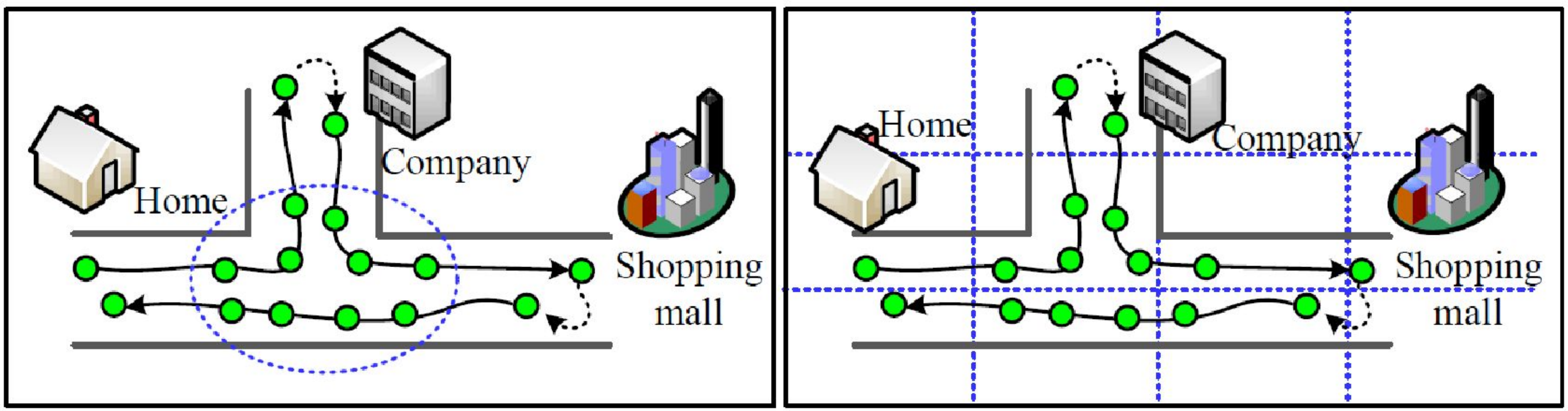
- DBSCAN algorithm: epsilon and MinPts



Density reachable

Mobility data processing

- Why simple density-based clustering does not work?

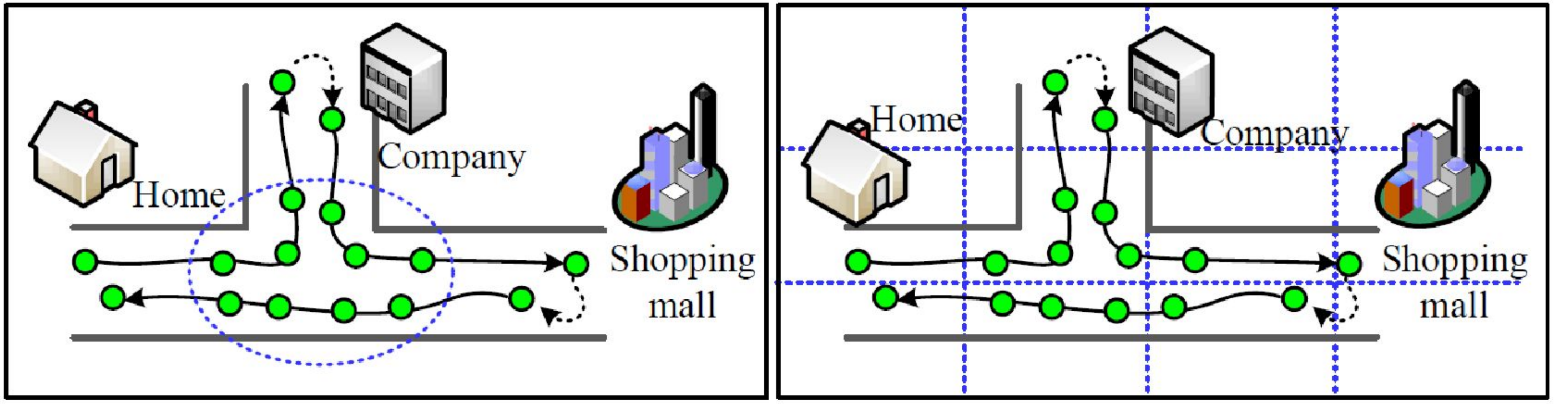


A) Clustering-based detection B) Grid-based detection

Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, and Xing Xie. 2009. Mining Individual Life Pattern Based on Location History. In Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware (MDM '09).

Mobility data processing

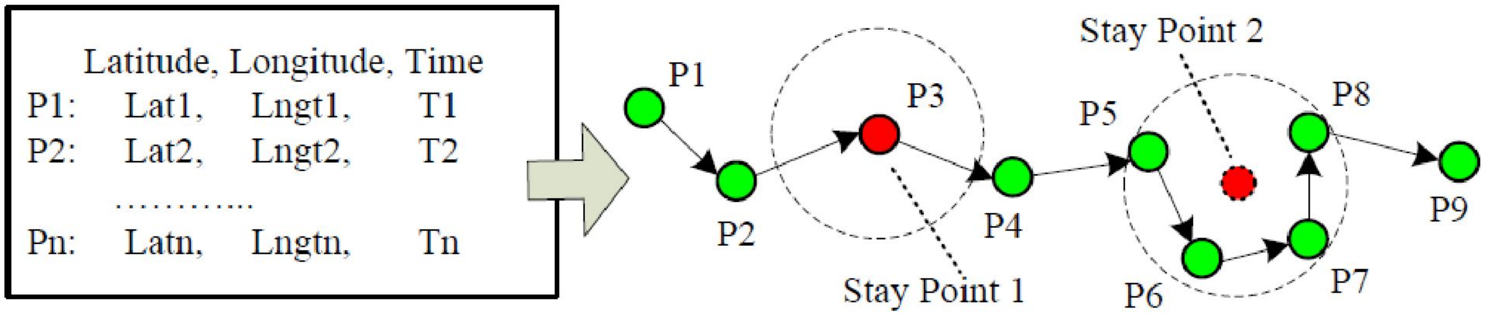
- Why density-based clustering does not work well?



A) Clustering-based detection **B) Grid-based detection**

False positives: roads / intersections have many points, but they are not stay points

Mobility data processing



Staying (stationar y)

Wandering around within a bounded region

- Iteratively seek the spatial region in which the individual stays for a period over a threshold
- Example: **a stay point** is detected if the individual spends more than 30 minutes within a range of 200 meters

Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, and Xing Xie. 2009. Mining Individual Life Pattern Based on Location History. In Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware (MDM '09).

Overview

- Previously: data collection & pre-processing (e.g., removing noise)
- Today: creating useful features
  - Time domain: Numerical, nominal, pattern mining
  - Frequency domain: Fourier analysis
  - Time + frequency: Wavelet analysis
  - Mobility data processing
  - Text data processing

Features for unstructured data

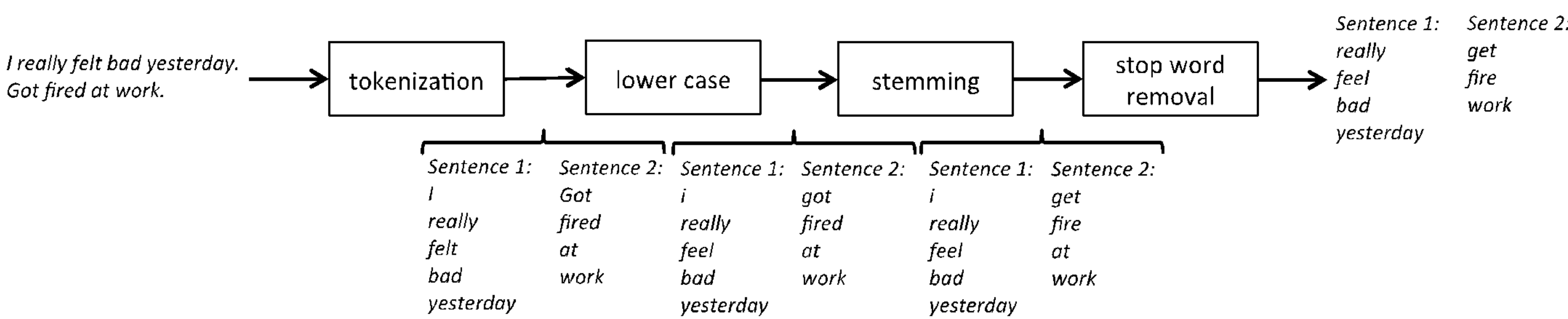
- Ok, we have seen a lot of possibilities for structured data
- How do we handle more unstructured data?
  - Free text
  - Images
  - Audio
  - Video
  - ……
- Look at text only in this lecture

Features for text (1)

- Let us take an example from Bruce:
  - Bruce: “I really felt bad yesterday. Got fired at work.”
  - Perform a number of steps first:
    - Tokenization
      - identify sentences and words within sentences
    - Lower case
      - change the uppercase letters to lowercase
    - Stemming
      - Identify the stem of each word to reduce words to their stem and map all different variations of for example verbs to a single term
    - Stop word removal
      - remove known stop words as they are not likely to be predictive

Features for text (2)

- Let us take an example from Bruce:
  - Bruce: ”I really felt bad yesterday. Got fired at work.”



Features for text (3)

- Approaches:
  - Bag of words
  - TF-IDF
  - Topic modeling
  - Embedding

Bag of words (1)

- Count occurrences of *n*-grams within text
- *n*-gram: *n* consecutive words
  - 2-gram (=bigram): “please turn”, “turn your”, or ”your homework”,
  - 3-gram (=trigram): “please turn your”, or “turn your homework”
- Assume that we note down the ***S* sentences** we have found within **an instance *i*** at for **attribute *j*** as follows:

$$\{x_i^j(1), \ldots, x_i^j(S)\}$$

- The words within the *first* sentence can be further indexed as:

$$\{x_i^j(1,1), \ldots, x_i^j(1,W)\}$$

Bag of words (2)

Algorithm 2: Bag of Words (n-grams)

$A = \{\}$
$N_{attr} = 1$
for $i = 1, \ldots, N$ **do** // for each instance *i*
  $a_i^1, \ldots, a_i^{N_{attr}} = 0$
  for $s = 1, \ldots, S$ **do** // for each sentence *s*
    for $w = 1, \ldots, W$ **do** // find *n*-gram
      if $w + (n-1) \leq W$ **then**
        $A_{temp} = < x_i^j(s, w), \ldots, x_i^j(s, w + (n-1)) >$
        if $A_{temp} \notin A$ **then**
          $A = A \cup A_{temp}$
          $a_i^{N_{attr}} = 1$
          $a_1^{N_{attr}}, \ldots, a_{i-1}^{N_{attr}} = 0$
          $N_{attr} = N_{attr} + 1$
        else
          $k = index(A_{temp})$
          $a_i^k = a_i^k + 1$
      end
    end
  end
end

a_i^j : the value of the attribute j for instance i
(# of occurrences of a given n-gram)

// find an n-gram that starts at position w
(here, only consider attribute j as denoted x_i^j)

N_attr is related to the number of words;
superscript denotes the k-th ngram discovered

TF-IDF (1)

- Bag of words does not account for the "uniqueness of words"
- For a given doc *i*, the number of occurrences of a word (using bag of words): a_i^j (here, doc i = sentence i)
  - This metric is known as Term frequency (TF): the number of occurrences of an n-gram in a given instance
- For all docs: {1, …, N}, the inverse document frequency is
  - How much information a word provides? Is it common or rare across all documents? IDF considers uniqueness!

N: the number of total instances N that contain the n-gram

$$idf_j = log\left(\frac{N}{|\{i|i \in \{1,\ldots,N\} \wedge a_i^j > 0\}|}\right)$$

Denominator: the number of docs (= sentences) whose $a_i^j > 0$

TF-IDF (2)

- And we multiply this IDF value with the number of times it occurs (TF):

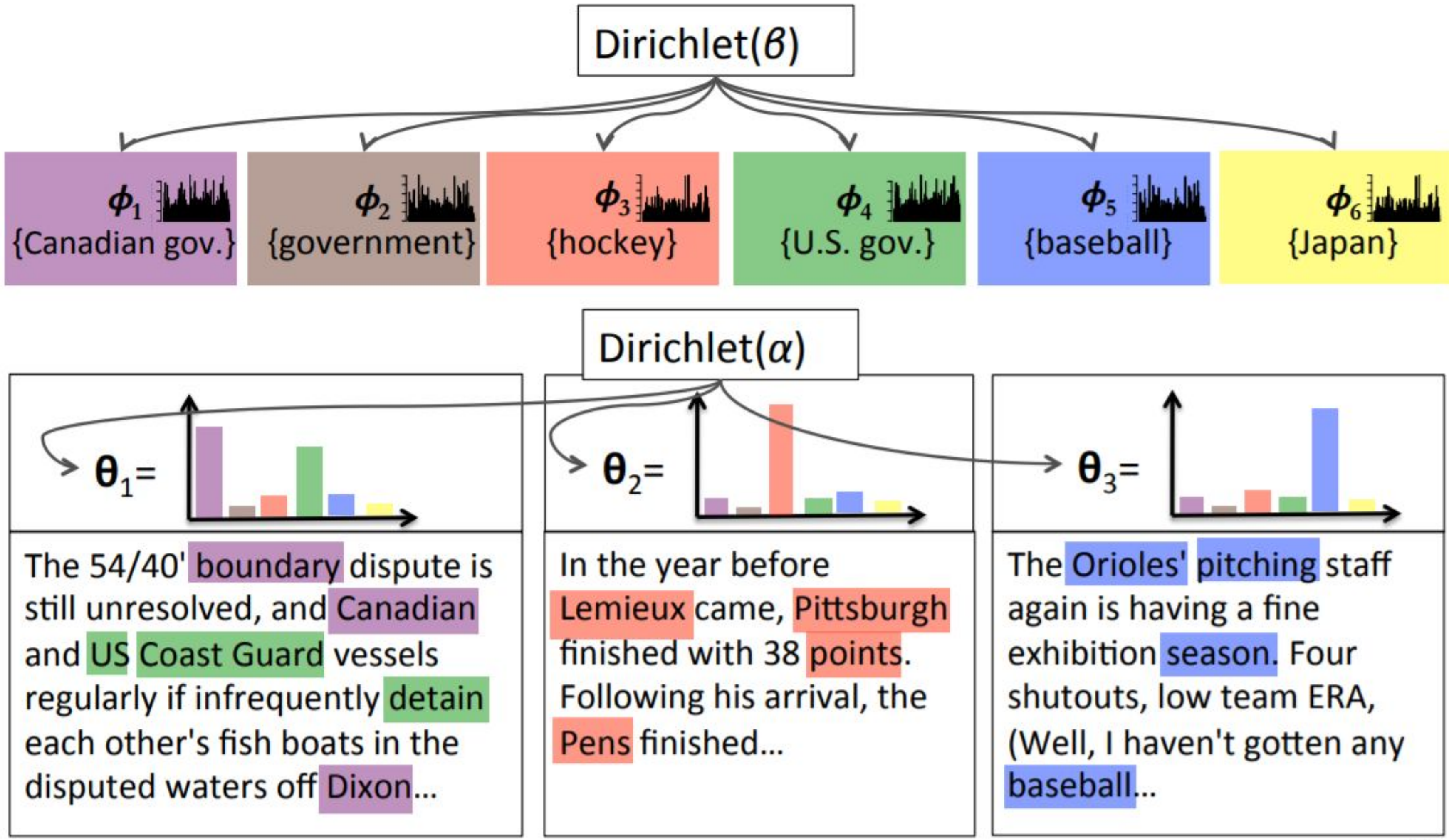
$$tf\_idf_i^j = a_i^j \cdot idf_j$$

- Key properties of TF-IDF:
  - Gives more weight to unique words
  - Avoids very frequent (and probably not very predictive words) to become too dominant

Topic Modeling (1)

- Instead of looking at words, let us look at the topics the free text is about
- We assume that the texts contain *k* topics (pre-set)
- Topics are associated with words, certain words make up a topic
  - The depression topic might contain words such as “bad”, “down”, “mood”, etc.
- For each topic, words have certain weights as follows:

$$topic(k) = \{< A_1, w_k^1 >, \ldots, < A_{N_{attr}}, w_k^{N_{attr}} >\}$$



https://www.cs.cmu.edu/~mgormley/courses/10701-f16/slides/lecture20-topic-models.pdf

Topic Modeling (2)

- How do we find topics?
  - Well known approach: Latent Dirichlet Allocation (LDA) (cf. Blei, 2003)
  - It assumes texts are generated with:
    - certain words in mind (using a Poisson distribution)
    - a distribution over topics (using a Dirichlet distribution)
  - Initially words are fully assigned to a single topic at random
  - Weights are updated to maximize the probability of observing the given texts
  - As a result, for a given topic, we have a set of words whose occurrences follows the multinomial probability distribution (e.g. “job” with a probability of 0.05 for the topic “work”)

Topic Modeling (3)

- Finding a numeric score to each topic
  - For a given instance or document (i), the score of topic *k* is given as the weighted sum of all the observed frequencies

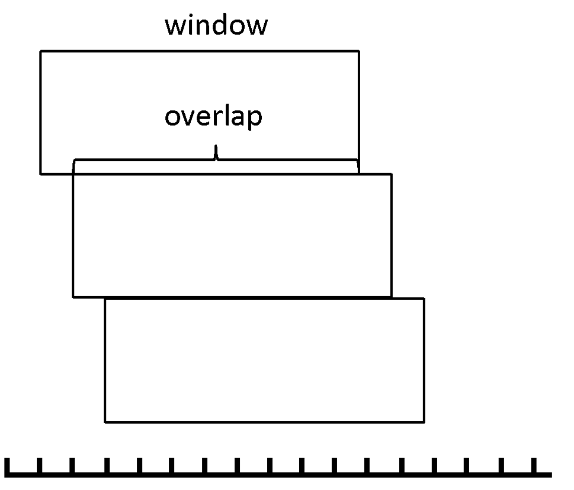
Topic k's weight for m-th word

$$topic_k(i) = \sum_{m=1}^{N_{attr}} a_i^m \cdot w_k^m$$

Observed frequency of m-th word in sentence/doc i

Composition of final dataset

- Moving windows (how much overlapping?)
  - Extreme case: If we have a large windows size, will it matter that we move one time point?
  - Disadvantages: overfitting because features are too similar (limited variation)
- Example
  - Typical: 50%, MSQL book: 90%
  - 90% overlapping
    - 2895 instances (out of 31,838)



Summary

- Time domain: Numerical, nominal, pattern mining
- Frequency domain: Fourier analysis
- Time + frequency: Wavelet analysis
- Mobility data processing
- Text data processing