Knowledge Discovery in Databases II

Lecture 3 – Data Streams

Prof. Dr. Peer Kröger, Yifeng Lu

Sommer Semester 2019

Credits:

Based on material of Eirini Ntoutsi, Matthias Schubert,

Arthur Zimek, Peer Kröger, Yifeng Lu

Table of Contents

Kapitel 1: Intorduction i

A Disclaimer (kind of ...)

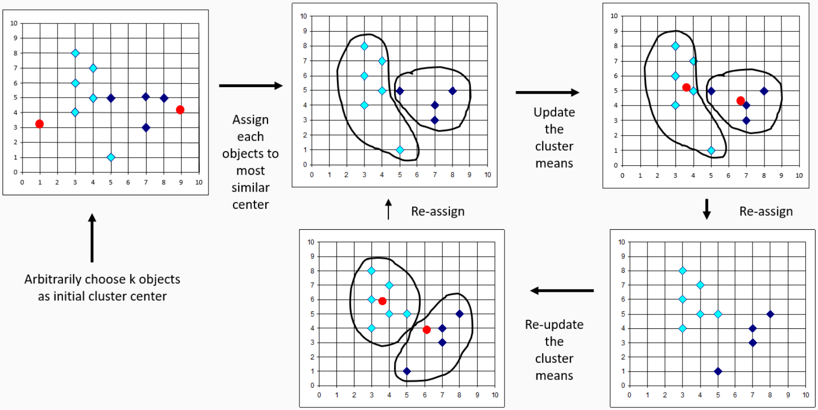
- Data streams usually are a very challenging source of data
- Analysis of data streams require to address several aspects such as
  - The hardware
  - The processing environment (like the operating system, the programming language and the programming schema, . . .)
  - The algorithmic design
  - . . .
- In this lecture, we focus on the algorithmic aspects that are necessary for processing data streams
- The lecture Big Data Management focuses on other aspects

Batch Learning

- Most of the DM algorithms focus on batch learning
  - The complete training/data set is available to the learning algorithm
  - Data instances can be accessed multiple times
  - e.g., for clustering: k-Means, DBSCAN
  - e.g., for classification: decision trees, Naïve Bayes
- Implict assumption: instances are generated by some stationary probability distribution; data is not volatile and so are patterns

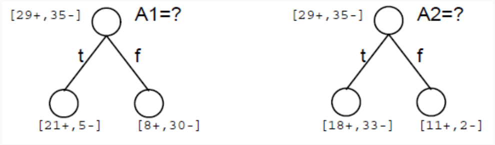
Example: Batch Clustering

- k-means (here $k = 2$) needs full access to the data in each iteration


Assign each objects to most similar center
Update the cluster means
Re-assign
Re-assign
Re-update the cluster means
Arbitrarily choose k objects as initial cluster center

Example: Batch Classification

- Decision Trees are constructed in a top-down recursive divide-and-conquer manner requiring full access to the data for each split
  - At start, all the training examples are at the root node
  - Select the best attribute for the root
  - For each possible value of the test attribute, a descendant of the root node is created and the instances are mapped to the appropriate descendant node
  - Repeat the splitting attribute decision for each descendant node, so instances are partitioned recursively

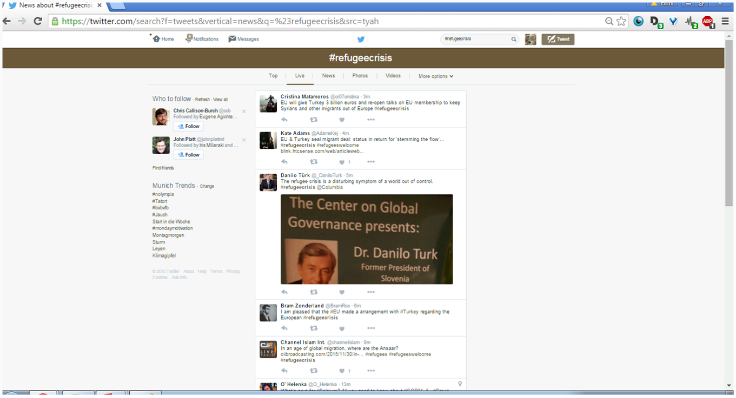


From Batch to Streams

- Many interesting applications nowadays come from dynamic environments where data are generated over time, e.g., customer transactions, call records, customer click data, social media interactions
- Batch learning is not sufficient anymore as
    - Data is never ending. What is the training set?
    - Multiple access to the data is not possible or desirable
- And also, the data generation process is subject to changes over time
    - The patterns extracted upon such sort of data are also evolving
    - Algorithms should respond to change (incorporate new data instances, forget obsolete data instances)

Examples

- Twitter stream for hastag “#refugeecrisis”



Who to follow · Refresh · View all

Chris Callison-Burch @ccb
Followed by Eugene Agichte…

John Platt @johnplattml
Followed by Iris Miliaraki and …

Find friends

Munich Trends · Change

#nolympia
#Tatort
#bvbvfb
#Jauch
Start in die Woche
#mondaymotivation
Montagmorgen
Sturm
Leyen
Klimagipfel

© 2015 Twitter About Help Terms Privacy Cookies Ads info

Cristina Matamoros @cr07cristina · 3m
EU will give Turkey 3 billion euros and re-open talks on EU membership to keep Syrians and other migrants out of Europe #refugeecrisis

Kate Adams @AdamsKej · 4m
EU & Turkey seal migrant deal: status in return for 'stemming the flow'… #refugeecrisis #refugeeswelcome blink.htcsense.com/web/articleweb…

Danilo Türk @_DaniloTurk · 5m
The refugee crisis is a disturbing symptom of a world out of control. #refugeecrisis @Columbia

The Center on Global Governance presents:
Dr. Danilo Turk
Former President of Slovenia

Bram Zonderland @BramRoc · 6m
I am pleased that the #EU made a arrangement with #Turkey regarding the European #refugeecrisis

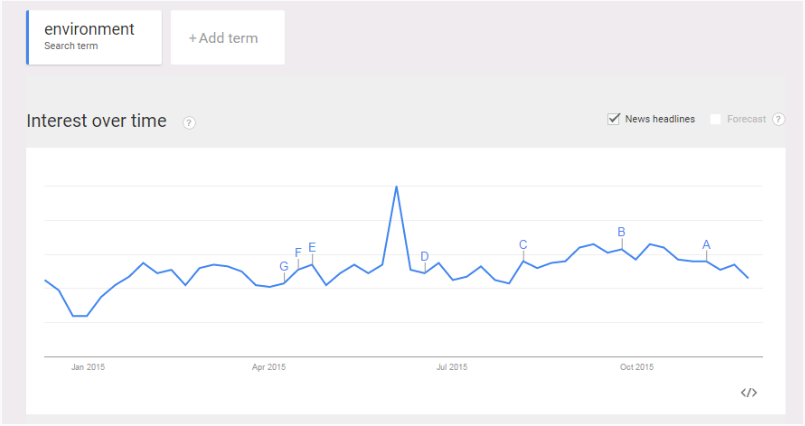
Channel Islam Int. @channelislam · 9m
In an age of global migration, where are the Ansaar? ciibroadcasting.com/2015/11/30/in-… #refugees #refugeeswelcome #refugeecrisis

O' Helenka @O_Helenka · 13m

Source: https://www.twitter.com/

Examples

- Trend of the search for “environment”



Source: `https://www.google.com/trends/`

Examples

- Experiments at CERN are generating an entire petabyte (1PB=106 GB) of data every second as particles fired around the Large Hadron Collider (LHC) at velocities approaching the speed of light are smashed together
- “We do not store all the data as that would be impractical. Instead, from the collisions we run, we only keep the few pieces that are of interest, the rare events that occur, which our filters spot and send on over the network”
- This still means CERN is storing 25PB of data every year — the same as 1,000 years’ worth of DVD quality video — which can then be analyzed and interrogated by scientists looking for clues to the structure and make-up of the universe

Source: `http://public.web.cern.ch/public/en/LHC/Computing-en.html`

Source: `http://www.v3.co.uk/v3-uk/news/2081263/cern-experiments-generating-petabyte`

Examples

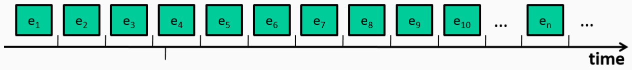
- Network monitoring records e.g. TCP connection records of LAN network traffic
- A connection is a sequence of TCP packets starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol
- Connections are described in terms of 42 features like duration, protocol type, service, flag, src bytes, dst bytes etc.
- Each connection is labeled as either normal, or as an attack, with exactly one specific attack type
- Most of the connections are usually normal, but occasionally there could be a burst of attacks at certain times

Source (with link to a real data set): `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`

What is a Data Stream?

Everything flows, nothing stands still

Heraclitus (535-475 BC)



- Data evolve over time as new data arrive (and old data become obsolete/irrelevant)
- We can distinguish between:
  - Dynamic data arriving at a low rate (as e.g. in DWs): incremental methods might work for such cases
  - Data streams: possible infinite sequence of elements arriving at a rapid rate: new methods are required to deal with the amount and complexity of these data

Incremental Methods

- Focus is on how to update the current pattern based on the newly arrived data, without re-computing the pattern from scratch
- Requires (limited) access to raw data (i.e., only the data that is affected by the changes)
- Example: incremental DBSCAN (insertion of a new point *p*)

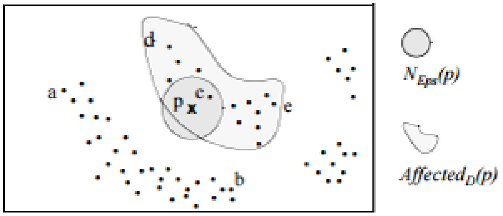


Figure 3: : Affected objects in a sample database

Challenges for Streams

- Data Mining over stream data is more challenging than batch learning
  - Huge amounts of data, thus, only a small amount can be stored in memory
  - Arrival at a rapid rate, thus, no much time for processing
  - The generative distribution of the stream might change over time rather than being stationary, thus, adapt and report on changes
- Requirements for stream mining algorithms
  - Use limited computational resources (bounded memory, small amount of available processing time)
  - No random access to the data but rather only one look at the data (upon their arrival)

From Data Changes to Pattern Changes

Example: cluster evolution over time

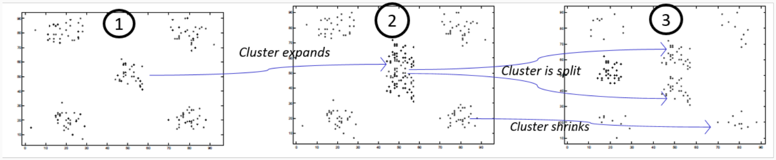


Figure: Data records at three consecutive time stamps, the clustering gradually changes
(from: *MONIC - Modeling and Monitoring Cluster Transitions*, Spiliopoulou et al, KDD 2006)

Example: decision boundary drift over time

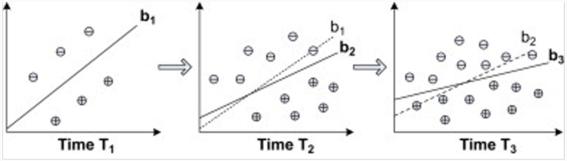


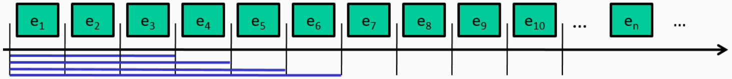
Fig. 1. An illustration of concept drifting in data streams. In the three consecutive time stamps T_1, T_2 and T_3, the classification boundary gradually drifts from b_1 to b_2 and finally to b_3.
(from: *A framework for application-driven classification of data streams*, Zhang et al, Journal Neurocomputing 2012)

Data Ageing

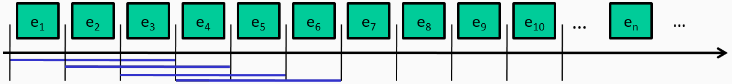
- Usually we are not interested in the whole history of the stream but only in the recent history
- There are different ageing/weighting mechanisms or window models that reflect which part of the stream history is important for learning
    - Landmark window model
    - Sliding window model
    - Damped window model

Data Ageing Models

- Landmark (window) model
  - Include all objects from a given landmark
  - All points have an equal weight (usually $w = 1$)

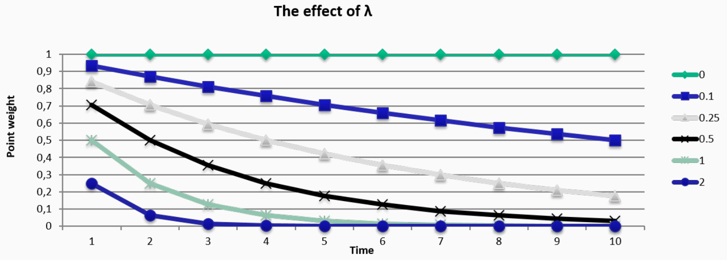


- Sliding window model
  - Remember only the *n* most recent entries, where *n* is the window size
  - All points within the window have a weight $w = 1$, for the rest: $w = 0$



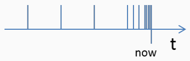
Data Ageing Models

- Damped window model
  - Data are subject to ageing according to a fading function $f(t)$, i.e., each point is assigned a weight that decreases with time t via $f(t)$
  - A widely used fading function in temporal applications is the exponential fading function: $f(t) = 2^{-\lambda t}$, where $\lambda > 0$ is the decae rate that determines the importance of historical data (the higher the value of λ, the lower the importance of old data)



Time Frames

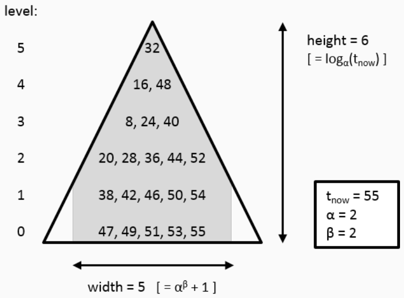
- Task: maintain the history of the stream
  - Store snapshots at (regular) time intervals
  - Use finer granularity for recent data for a detailed representation
  - Use coarser granularity for older data to save space



- Tilted time frame (tilt time frame)
  - Example: align time axis with natural calendar time, e.g.:
    1 snapshot per minute for the 15 most recent minutes
    1 snapshot per quarter for the 4 most recent quarters
    1 snapshot per hour for the 24 most recent hours
    1 snapshot per day for the 30 most recent day
    1 snapshot per month for the 12 most recent months
    Total number of snapshots for one year: 85
    (compare to 60*24*30*12 = 518400 snapshots)

Pyramidal Time Frame Model

- Stores snapshots in levels of decreasing cardinality (pyramid)
- Size (number of snapshots) is controlled by two parameters $\alpha, \beta \in \mathbb{N}$



- For a new snapshot *A* provided at time t_{now}
  - Store *A* on the highest level *i* with $t_{now} mod \alpha^i = 0$
  - If a level contains more than $\alpha^\beta + 1$ snapshots, remove the oldest
- Maximal height $\lceil \log_\alpha(t_{now}) \rceil$
- Number of snapshots is smaller than $(\alpha^\beta + 1) \cdot \lceil \log_\alpha(t_{now}) \rceil$
- Example: 1 snapshot per second for 100 years using $\alpha = 2$ and $\beta = 1$ results in 96 snapshots

Kapitel 2: Clustering i

Overview

- The (batch) clustering problem
    - Given a set of measurements, observations, etc., the goal is to group the data into groups of similar data objects (clusters)
- The data stream clustering problem
    - Continuously maintain a consistently good clustering of the sequence observed so far, using a small amount of memory and time
- This implies
    - Use incremental computations and techniques
    - Maintaining cluster structures that evolve over time
    - Working with summaries (of such cluster structures) instead of raw data

Challenges

- Traditional clustering methods require access upon the whole data set
- Rather, we need online maintenance of patterns that captures pattern drifts
- The underlying population distribution might change: drifts/ shifts of concepts
- One clustering model might not be adequate to capture the evolution
- The role of outliers and clusters are often exchanged in a stream
- A clear and fast identification of outliers is often crucial for the success

Overview

Cluster Model	Batch/static clustering	Dynamic/stream clustering
Partitioning methods	k-means, k-medoid	– Leader – STREAM k-Means – CluStream
Density-based methods	DBSCAN, OPTICS	– DenStream – incDBSCAN – incOPTICS
Grid-based methods	STRING	– Dstream

Overview

- Goal: Construct a partition of a set of objects into k clusters
- Two types of methods
  - Adaptive methods such as Leader (Spath 1980), Simple single pass k-Means (Farnstrom et al., 2000), STREAM k-Means (OCaEtAl02)
  - Online summarization - offline clustering methods such as CluStream (AggEtAl03), DenStream (CaoEtAl06)
  - Continous grid-based such as DStream (CheTu07)

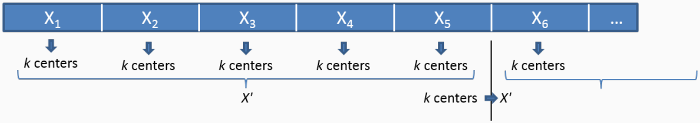
Kapitel 2: Clustering

Leader

- The simplest single-pass partitioning algorithm
- Whenever a new instance p arrives from the stream
    - Find its closest cluster (leader), c_{clos}
    - Assign p to c_{clos} if their distance is below the threshold d_{thresh}
    - Otherwise, create a new cluster (leader) with p
- Properties
    - 1-pass and fast algorithm
    - No prior information on the number of clusters required
    - Result depends on the order of the examples
    - Sensitive to a correct guess of d_{thresh} (which is fixed)

STEAM k-Means

- Simple extension of batch *k*-Means to streams:
  - Use a buffer (chunk) that fits in memory and apply *k*-Means locally in the buffer
- STEAM *k*-Means:
  - Apply *k*-Means on chunk X_i
  - X' denotes the set of $i \cdot k$ cluster centers from all chunks $X_1, \ldots X_i$ each weighted by the number of points assigned to it
  - Output the *k* centers obtained by clustering X'



STEAM k-Means

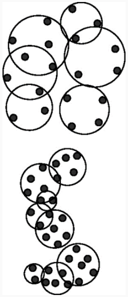
Properties:

- Pros:
  - Single scan
- Cons:
  - Expensive (according to authors)
  - No aging
  - Cluster model inherent limitations (no noise handling, ...)
  - Fixed *k* in all chunks

Kapitel 2: Clustering

Basic Idea

- Online component
    - Maintain a larger number of small clusters (micro-cluster)
    - Reduce data, keep sufficient details
    - Separate clusters for noise (improved robustness)
    - Provide accurate and fine grained input for further steps
- Offline component
    - Generate actual clustering on user request using micro-cluster information
    - Exchangeable clustering method
    - Individual and changing parameterization possible
    - Only approximate clustering



Micro Clusters: Cluster Features

- Clustering Features[1] for a set of points X: $CF_X = (N_X, LS_X, SS_X)$ with
  - N_X is the number of points, i.e., $|X|$
  - LS_X is the linear sum of all points in X, i.e., $\sum_{x_i \in X} x_i$
  - SS_X is the squared sum of all points in X, i.e., $\sum_{x_i \in X} x_i^2$
- From CF_X we can easily compute basic statistics of X such as
  - Mean (centroid) of X
  - Compactness measures such as radius, diameter, variance and std. deviation
- CFs are additive, i.e., given two (disjunctive) sets X and Y with their corresponding CF_X and CF_Y, we can compute $CF_{X \cup Y}$ as follows:

$$CF_{X \cup Y} = CF_X + CF_Y = (N_X + N_Y, LS_X + LS_Y, SS_X + SS_Y)$$

[1] Zhang, Ramakrishnan, Linvy: BIRCH: An Efficient Data Clustering Method for Very Large Databases. Proc. ACM SIGMOD 1996

Micro Clusters: Data Bubbles

- While CFs are good for partitioning based clustering, they do not capture density estimations necessary for e.g. OPTICS
- Data Bubbles[2] for a set of points X: $B_X = (N_X, M_X, r_X)$ with
    - N_X is the number of points, i.e., $|X|$
    - M_X is the centroid of X
    - r_X is the radius of the ball centered at M capturing all points in X
- Data Bubbles can be computed from CFs
- Data Bubbles allow a good approximation of core/reachability distances for hierarchical clustering

[2] Breunig, Kriegel, Kröger, Sander: Data Bubbles: Quality Preserving Performance Boosting for Hierarchical Clustering. Proc. ACM SIGMOD 2001

CluStream — Basics

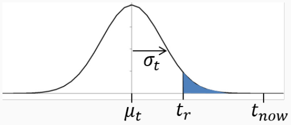
- One of the first algorithms for streams proposing an online/offline framework
- Uses cluster features to propose a k-Means like stream clustering method
- Cluster Features (see above) are extended by the information of the time slots T when points in X have arrived, i.e. x_i has arrived at time t_i:
  $CFT_X = (N_X, LS_X, SS_X, LST_X, SST_X)$, where
  - N, LS_X, and SS_X are defined as above (note that LS_X and SS_X are vectors)
  - LST_X is the linear sum of time slots of X, i.e., $\sum_{t_i \in T} t_i$
  - SST_X is the linear sum of time slots of X, i.e., $\sum_{t_i \in T} t_i^2$
- Again, important for the stream situation:
  - CFTs can be maintained incrementally, i.e. $CFT_{X \cup p} = CFT_X + p$

CluStream — Method at a Glance

- General idea: a fixed number of *q* micro-clusters (represented as CFTs) is maintained over time
- Initialize: apply *q*-Means over a buffer of *initP* observations and build a summary for each cluster
- Both *q* and *initP* are input parameters
- Upon request, *k*-Means can be applied to a snapshot of the *q* CFTs

CluStream — Online Phase

- Maintain q micro-clusters while adding a new observation x_i from the stream
  - Find closest micro-cluster MC_j according to distance $dist(x_i, \mu_j)$
  - If $dist(x_i, \mu_j) < \alpha \cdot \sigma_j$ then add x_i to MC_j
  - Else create a new micro-cluster containing only x_i and delete a micro cluster by using one of the following actions:
    - Delete the least recent MC if its relevance stamp $t_r < t_{now} - \tau$
    - Merge the two closest micro clusters
- $\alpha \cdot \sigma_j$ is called the maximal boundary of MC_j
- The relevance stamp t_r of MC_j approximates the average time stamp of the last m objects
- It is computed as the time of arrival of the $m/(2 \cdot N)$-th percentile (i.e., $1 - m/2 \cdot N$ of the time stamps in MC_j



CluStream — Offline Phase

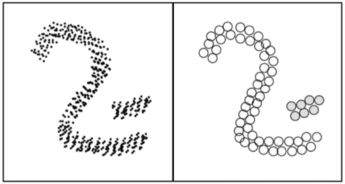
- Snapshots of micro-clusters are stored in pyramidal time frame
- Given k and a time horizon h
- Locate all valid micro-clusters within h
- Final clusters are gained using a modified k-Means
    - Micro-clusters over a certain time horizon are treated as pseudo-points
    - In the initialization: seeds are not picked randomly, but sampled with a probability proportional to N
    - Distances are calculated between centroids of the micro-clusters
    - New seeds are weighted by N
    - The k clusters obtained from applying k-Means on the micro-clusters are called macro-clusters

CluStream — Properties

- Single scan, stream compression using micro-clusters
- Views the stream as a changing process over time, rather than clustering the whole stream at a time
- Can characterize clusters over different time horizons in changing environment
- Aging only for entire clusters
- Noise handling offline
- Not adaptive q is fixed, requires $k < q$
- Many parameters
- Sensitive to outliers and noise (also model inherent)

DenStream — Motivation

- Density-base cluster model: clusters as regions of high density surrounded by regions of low density (noise)
- Very appealing for streams
    - No assumption on the number of clusters
    - Discovering clusters of arbitrary shapes
    - Ability to handle outliers and noise
- But, they miss a clustering model (or it is too complicated): clusters are represented by all their points
- So we can again only hope to approximate an arbitrary shaped cluster by many small (circular) micro-clusters



DenStream — Basics

- The DenStream algorithm uses time-weighted cluster features at time slot t given a time weighting function f for observations x_i arriving at time $t_i < t$:

$$CF_X^t = (N_X^t, LS_X^t, SS_X^t)$$

  where

  - $N_X^t = \sum_{x_i \in X} f(t - t_i)$
  - $LS_X^t = \sum_{x_i \in X} f(t - t_i) x_i$
  - $SS_X^t = \sum_{x_i \in X} f(t - t_i) x_i^2$
- Usually, $f(t) = 2^{-\lambda t}$ models the damped window model (but other functions are possible)

DenStream — Basics

- If a new observation x_i is added, a micro-cluster summary CF_X^t can be maintained incrementally (analogously as above)
- If no point is added to CF_X^t for time interval Δt, then $CF_X^t = (2^{-\lambda\Delta t} \cdot N, 2^{-\lambda\Delta t} \cdot LS_X^t, 2^{-\lambda\Delta t} \cdot SS_X^t)$
- The radius r_X of a micro-cluster X can be derived from the cluster feature CF_X^t as follows

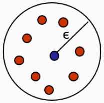
$$r_X = \sqrt{SS_X^t / N_X^t - (LS_X^t / N_X^t)^2}$$

- Analogously, the center c_X of a micro-cluster can be computed from its CF_X^t

DenStream — Basics

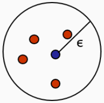
Given the density threshold μ (#points) and ε (volume) and a weighting factor β ($0 < \beta \leq 1$), DenStream maintains three different types of micro-clusters:

- Core (or dense) micro-clusters (CMC) X if
  $N_X^t \geq \mu$ and $r_X \leq \varepsilon$

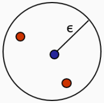


- Potential core micro-clusters (PCMC) X if
  $N_X^t \geq \beta \cdot \mu$ and $r_X \leq \varepsilon$

  (provides the opportunity for transitions between new clusters and outliers)



- Outlier micro-clusters (OMC) X if
  $r_X \leq \varepsilon$ and $N_X^t < \beta \cdot \mu$



Note: all MC types always have a radius $\leq \varepsilon$

DenStream — Initialization

- Collect a set I of *initP* of initial points
- For any $p \in I$:
    - Compute ε-neighborhood $N_\varepsilon(p)$ of p
    - If $|N_\varepsilon(p)| \geq \mu$ (p is core), create a new CMC $X = N_\varepsilon(p)$ and remove X from I
- For all remaining $p \in I$: create a new OMCs $X = N_\varepsilon(p)$ and remove X from I

DenStream — Online Phase

Online micro-cluster maintenance (when a new observation x_i arrives)

- Core micro-clusters are not considered
- Find closest potential core micro-cluster X_p
- If $dist(x_i, c_{X_p}) \leq \varepsilon$
  - Add x_i to X_p
  - Check if X_p becomes a CMC
- Else
  - Find closest outlier micro-cluster X_o
  - If $dist(x_i, c_{X_o}) \leq \varepsilon$, add x_i to X_o and check if X_o becomes a PCMC
  - Else: create a new OMC $X_{x_i} = \{x_i\}$
- After a given number of T time steps, check:
  - Delete all CMC X with $N_X^t < \mu$
  - Delete all OMC that did not become CMC within the last T time steps

DenStream — Offline Phase

- Upon user request, run DBSCAN on current CMCs and PCMCs
- Use centers and weights of the micro-clusters

DenStream — Discussion

- Single scan, stream compression using micro-clusters
- Noise/ outlier handling (model inherent)
- Flexible data aging model (for individual objects)
- Constant parameters over time, what about clusters with changing density?

Online/Offline-Approaches: Summary

	CluStream	DenStream
Online	convex micro cluster	
Offline	k-Means	DBSCAN
Aging	entire MCs	individual objects

- Cluster algorithm in offline phase exchangeable in principle
- Still “high” online costs (check all MCs)
- Many variants exist

Kapitel 2: Clustering

Basic Idea

- A grid structure is used to capture the density of the data set
- A cluster is a set of connected dense cells (see e.g. STING)
- Appealing features
    - No assumption on the number of clusters
    - Discovering clusters of arbitrary shapes
    - Ability to handle outliers
- In case of streams
    - The grid cells are considered as micro-clusters, i.e., summary information on cells are maintained
    - Update these summaries on the grid structure as the stream proceeds
    - Sample method: DStream (CheTu07)

DStream — Basics

- DStream divides each dimension into l partitions resulting in l^d cells (d: data dimensionality)
- Populated grid cells are maintained in a hash list
- For a grid cell C, the following summary is stored:

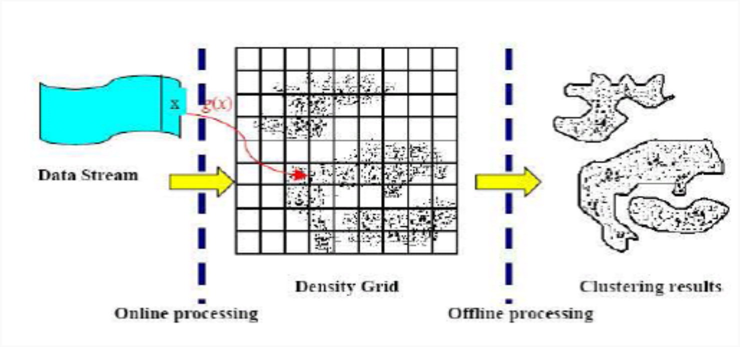
$$CF_C = (t_{update}, t_{spor}, N_C, label_C, status_C)$$

where

  - t_{update} is the last update time
  - t_{spor} last time, C has been removed
  - $N_C = \sum_{x_i \in C} \lambda^{t-t_i} \cdot x_i$ (count using damped window aging)
  - $label$ is the cluster label
  - $status \in \{sporadic, normal\}$

DStream — Overview

- DStream follows the online/offline paradigm
- Online mapping of the new data into the grid
- Offline computation of grid density and clustering of dense cells



DStream — Summaries

- Three cell types are defined by parameters τ_{dense} and $\tau sparse$:
  - Cell C is dense if $N_C > \tau_{dense}$
  - Cell C is sparse if $N_C < \tau_{sparse}$
  - Cell C is transitional if $\tau_{sparse} < N_C < \tau_{dense}$
- Connected regions of dense or transitional cells form a cluster
- Changes of the *status* occur in the online component
  - Set status to *normal*, if C changed from *sparse* to another type
  - Set status to *sporadic*, if for C the number of insertions into C is less than expected since the last update

DStream — Online Phase

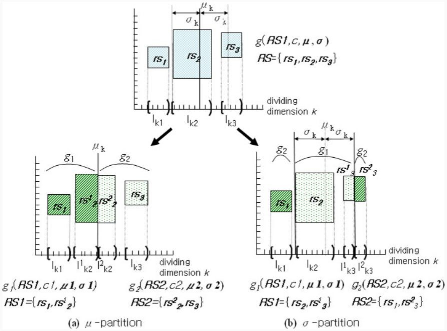
- Online grid cell maintenance (for new observation o_i):
  - Determine the grid cell C that x_i falls into
  - Add C to the hash list if it is not already contained
  - Update CF_C w.r.t. x_i and set status to normal if type changed from sparse
  - Periodically after T time steps
    - Delete all grid cells from the hash list that have been marked as *sporadic* and did not receive new points within the last T time steps
    - Mark sparse grid cells as *sporadic* if requirements (see previous slide) are met
    - Adjust the clustering

DenStream — Discussion

- Single scan, stream compression using micro-clusters
- Noise/ outlier handling (model inherent)
- Aging model for entire cells
- Constant parameters over time, what about clusters with changing density?
- Curse of dimensionality (number of grid cells is l^d)

Grid-based Variant: Dynamic Splitting

- Initially generate coarse grid
- Online split cells with interval $> \lambda$:
  - Mean split in the dimension with maximal variance
  - Around mean segment in the dimension with minimal variance

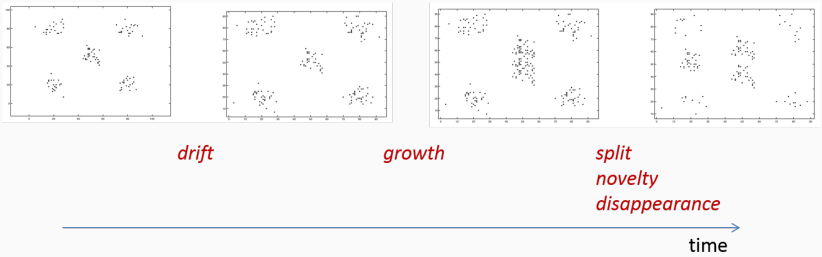


- Delete outdated cells based on user defined threshold (initial grid cells are never deleted)
- No particular offline component

Kapitel 2: Clustering

Basic Idea

- Detect and differentiate different types of changes
  - Disappearance of concepts
  - Migration/changes/drift of concepts
  - Merging of existing concepts
  - Splitting of groups vs. newly emerging clusters



MONIC: Modeling and Monitoring Cluster Transitions

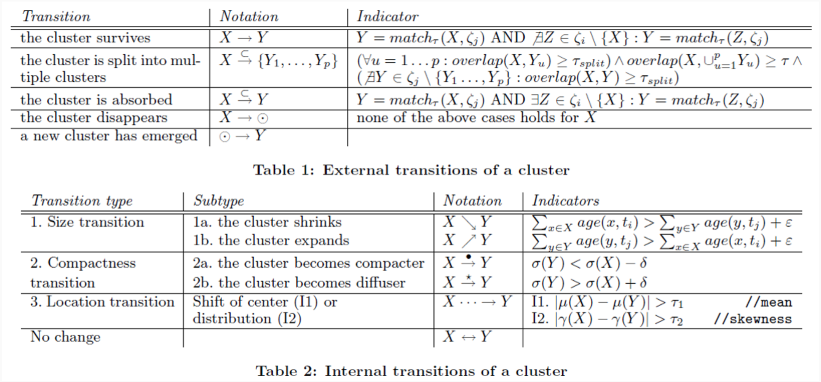
- MONIC[3] does not assume a particular cluster model
- Cluster matching for different points in time
  - Let X be a cluster at t_X and Y a cluster from the set of clusters ζ_Y at a later slot $t_Y > t_X$ and let the overlap between X and Y be

$$\mathit{overlap}(X, Y) = \frac{\sum_{o \in X \cap Y} \mathit{age}(t_Y, o)}{\sum_{x \in X} \mathit{age}(t_Y, x)}$$

  - Y is a match for X, $\mathit{match}_\tau(X, \zeta_Y) = Y$ subject to a threshold $\tau \in [0.5, 1]$ if Y is the cluster with the maximum overlap of at least τ where the overlap between two clusters X and Y is
  - If there is no cluster in the clustering ζ_Y at t_Y with an overlap of at least τ, then $\mathit{match}_\tau(X, \zeta_Y) = \emptyset$
  - The matching is not unique: several old clusters can be matched with the same new cluster

[3] Spiliopoulou et al.: MONIC - Modeling and Monitoring Cluster Transitions. Proc. KDD'06

MONIC: Cluster Transitions



Transition	*Notation*	*Indicator*
the cluster survives	$X \to Y$	$Y = match_\tau(X, \zeta_j)$ AND $\nexists Z \in \zeta_i \setminus \{X\} : Y = match_\tau(Z, \zeta_j)$
the cluster is split into multiple clusters	$X \overset{\subseteq}{\to} \{Y_1, \ldots, Y_p\}$	$(\forall u = 1 \ldots p : overlap(X, Y_u) \geq \tau_{split}) \wedge overlap(X, \cup_{u=1}^{p} Y_u) \geq \tau \wedge$ $(\nexists Y \in \zeta_j \setminus \{Y_1 \ldots, Y_p\} : overlap(X, Y) \geq \tau_{split})$
the cluster is absorbed	$X \overset{\subseteq}{\to} Y$	$Y = match_\tau(X, \zeta_j)$ AND $\exists Z \in \zeta_i \setminus \{X\} : Y = match_\tau(Z, \zeta_j)$
the cluster disappears	$X \to \odot$	none of the above cases holds for X
a new cluster has emerged	$\odot \to Y$	

Table 1: External transitions of a cluster

Transition type	*Subtype*	*Notation*	*Indicators*
1. Size transition	1a. the cluster shrinks	$X \searrow Y$	$\sum_{x \in X} age(x, t_i) > \sum_{y \in Y} age(y, t_j) + \varepsilon$
	1b. the cluster expands	$X \nearrow Y$	$\sum_{y \in Y} age(y, t_j) > \sum_{x \in X} age(x, t_i) + \varepsilon$
2. Compactness transition	2a. the cluster becomes compacter	$X \overset{\bullet}{\to} Y$	$\sigma(Y) < \sigma(X) - \delta$
	2b. the cluster becomes diffuser	$X \overset{\star}{\to} Y$	$\sigma(Y) > \sigma(X) + \delta$
3. Location transition	Shift of center (I1) or distribution (I2)	$X \cdots \to Y$	I1. $\lvert \mu(X) - \mu(Y) \rvert > \tau_1$ //mean I2. $\lvert \gamma(X) - \gamma(Y) \rvert > \tau_2$ //skewness
No change		$X \leftrightarrow Y$	

Table 2: Internal transitions of a cluster

Source: Spiliopoulou et al.: MONIC - Modeling and Monitoring Cluster Transitions. Proc. KDD'06

Summary: Stream Clustering

- A very important task given the availability of streams nowadays
- Stream clustering algorithm maintain a valid clustering of the evolving stream population over time
- Two generic approaches
    - Online maintenance of a final clustering model
    - Online summarization of the stream and offline clustering
- Different window models
- Evaluation is not straightforward (existing measures mostly for static case)
- Specialized approaches for text streams, high-dimensional streams.

Selected Readings on Stream Clustering

- C. Aggarwal: Data Streams: Models and Algorithms. Springer, 2007
- C. C. Aggarwal, J. Han, J. Wang, P. S. Yu: A framework for clustering evolving data streams. Proc. VLDB, 2003
- M. Ester, H.-P. Kriegel, J. Sander, M. Wimmer, X. Xu: Incremental Clustering for Mining in a Data Warehousing Environment. Proc. VLDB 1998
- J. Gama: Knowledge Discovery from Data Streams. Chapman and Hall/CRC, 2010
- F. Cao, M. Ester, W. Qian, A. Zhou: Density-Based Clustering over an Evolving Data Stream with Noise. Proc. SDM 2006
- Y. Chen, L. Tu: Density-Based Clustering for Real-Time Stream Data. Proc. KDD, 2007
- F. Farnstrom, J. Lewis, C. Elkan: Scalability for clustering algorithms revisited. ACM SIGKDD Expl. 2(1):51-57, 2000
- S. Guha, A. Meyerson, N. Mishra, R. Motwani, L. O' Callaghan: Clustering data streams: Theory and practice. IEEE TKDE 15(3):515–528, 2003
- L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, R. Motwani: Streaming-Data Algorithms for High-Quality Clustering. Proc. ICDE, 2002

Kapitel 3: Classification i

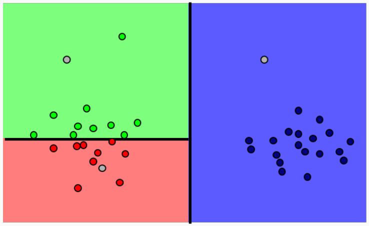
Kapitel 3: Classification

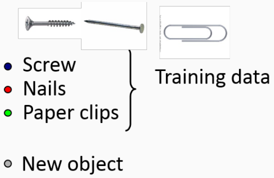
Classification Task

Recall: the classification task is to learn from the already classified training data, the rules to classify new objects based on their characteristics

The result attribute (class variable) is nominal (categorical)

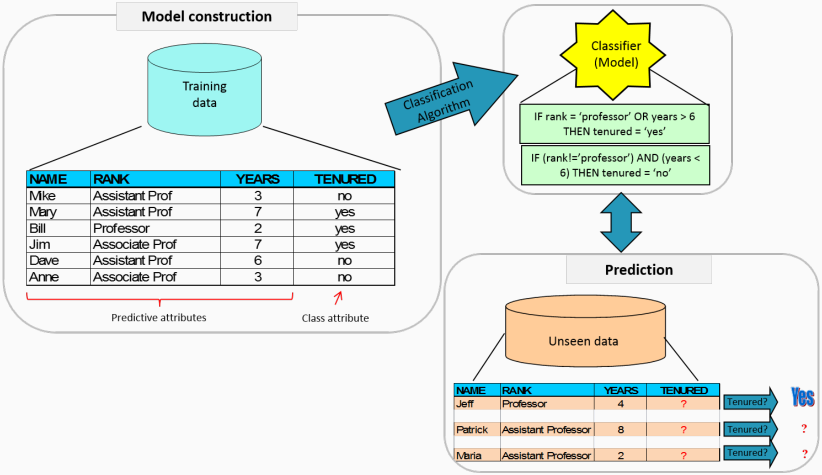
Example:





Classification Task

The batch classification process:



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

IF rank = 'professor' OR years > 6 THEN tenured = 'yes'

IF (rank!='professor') AND (years < 6) THEN tenured = 'no'

NAME	RANK	YEARS	TENURED
Jeff	Professor	4	?
Patrick	Assistant Professor	8	?
Maria	Assistant Professor	2	?

Stream versus batch classification

- So far, classification as a batch/ static task
  - The whole training set is given as input to the algorithm for the generation of the classification model.
  - The classification model is static (does not change)
  - When the performance of the model drops, a new model is generated from scratch over a new training set
- But, in a dynamic environment data change continuously
  - Batch model re-generation is not appropriate/sufficient anymore
- Rather: new classification algorithms required that
  - have the ability to incorporate new data
  - deal with non-stationary data generation processes (concept drift) and are able to remove obsolete data
  - are subject to resource constraints (processing time, memory)
  - use only a single scan of the data (one look, no random access)

Non-stationary data distribution

- In dynamically changing and non-stationary environments, the data distribution might change over time yielding the phenomenon of concept drift
- Different forms of change
  - The input data characteristics might change over time
  - The relation between the input data and the target variable might change over time
- Given the joint probability distribution $p_t(x_i, y)$ between instance x_i and class y at time slot t, the concept drift between time stamps t_0 and t_1 can be defined as

$$\exists x_i : p_{t_0}(x_i, y) \neq p_{t_1}(x_i, y)$$

Concept drift

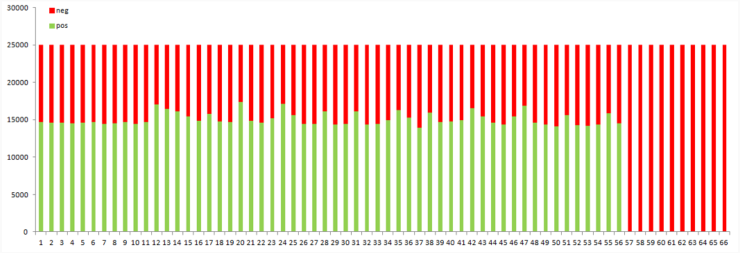
- According to the Bayesian Decision Theory

$$p(y|x_i) = \frac{p(y)p(x_i|y)}{p(x_i)}$$

- So changes in data can be characterized as changes in
    - The prior probabilities of the classes $p(y)$
    - The class conditional probabilities $p(x_i|y)$
    - The posterior $p(y|x_i)$ might change

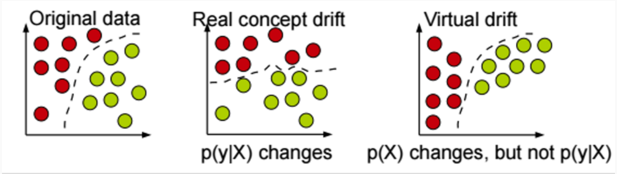
Evolving class priors

- The class distribution might change over time
- Example: Twitter sentiment data set
  - 1.600.000 instances split in 67 chunks of 25.000 tweets per chunk
  - Balanced data set (800.000 positive, 800.000 negative tweets)
  - The distribution of the classes changes over time
  - Dataset online at: `https://sites.google.com/site/twittersentimenthelp/for-researchers`



Real Drift and Virtual Drift

- Real concept drift
    - Refers to changes in $p(y|x_i)$
    - Such changes can happen with or without change in $p(x_i)$
    - E.g., “I am not interested in tech posts anymore”
- Virtual concept drift
    - If the $p(x_i)$ changes without affecting $p(y|x_i)$

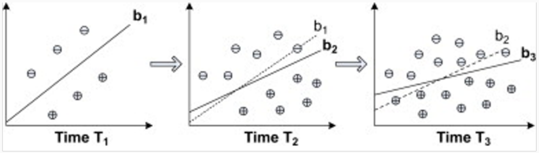


Source: [GamaETAl13]

- Drift (gradual changes) versus Shift (abrupt changes)

Real Drift and Virtual Drift

- As data evolve over time, the classifier should be updated to “reflect” the evolving data
  - Update by incorporating new data
  - Update by forgetting obsolete data



The classification boundary gradually drifts from b_1 (at T_1) to b_2 (at T_2) and finally to b_3 (at T_3).
(Source: A framework for application-driven classification of data streams, Zhang et al, Journal Neurocomputing 2012)

Data Stream Classifiers

- The batch classification problem
  - Given a finite training set $D = \{(x_i, y)\}$, where y is the class label, find a function $y = f(x)$ that can predict the y value for an unseen instance x
- The data stream classification problem
  - Given an infinite[4] sequence of pairs of the form (x_i, y), find a function $y = f(x)$ that can predict the y value for an unseen instance x
  - Assumption: the label y of x is not available during the prediction time but it is available "shortly" after for model update[5]

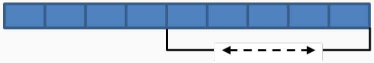
[4] Aging models are not so important for classification — Why?

[5] See also: the subsection on Evaluation later

Classification Using Chunks — Approaches

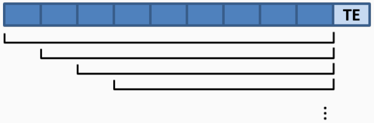
Train/Retrain a classifier at regular intervals (chunks)

- Window size can be fixed or adaptive



Train classifiers for several window sizes (chunks)

- Input: m chunks $X_1, \ldots, X_m$ of equal size $|X|$
- For $h = 1 \ldots m$: train a classifier on the most recent $h-1$ chunks
- Test the performance on the newest chunk
- Use best classifier



Classification Using Chunks — Approaches

Train a weighted set (ensemble) of k classifiers Cl_i from sequential chunks

- For a new chunk X_l
    - Train a new classifier using X_l as training data
    - Test all classifiers Cl_i using X_l as test data and set their weight to

      $$w_i = \max\{MSE_r - MSE_i, 0\}$$

      where $MSE_r = \sum_{y_j \in Y} p(y_j)(1-p(y_j))^2$ is the error of a random classifier and

      $$MSE_i = 1/|X_i| \cdot \sum_{x_l \in X_l, y_j \in Y} (1 - p_{CL_i}(y_j|x_l))^2$$

      is the error of classifier Cl_i
    - Keep the best k classifiers
- Classification: weighted average of the ensemble

Recurring concepts and noise (FLORA 3 and 4)

- The FLORA[6] system maintains decision lists per class
- Dealing with recurring contexts (Flora 3)
  - If the current concept (descriptor sets) is stable, store it for later reuse
  - If a drift is suspected, inspect stored concepts
    - Find the best candidate concept
    - “Refit” the concept (reset counters and re-process all examples in the window)
    - Evaluate the concept (choose the candidate if the resulting description is smaller)

  Note: old concepts are not just retrieved and used as the current concept but re-generalized with the examples of the current window
- Robustness against noise (Flora 4)
  - The decision lists are allowed to cover some positive or negative examples, resp.

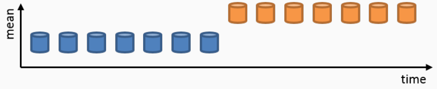
[6]Widmer et al.: Learning in the Presence of Concept Drift and Hidden Contexts. Machine Learning, 1996

Handling Concept Drift

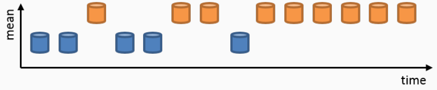
Types of changes formalized by Bifet et al.: Handling Concept Drift: Importance, Challenges & Solutions. Tutorial at PAKDD 2011

Speed:

– Sudden

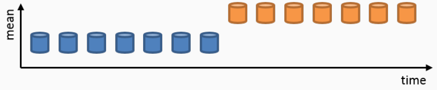


– Gradual

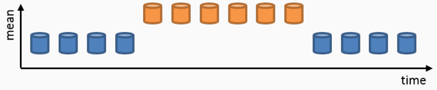


Reoccurrence:

– None

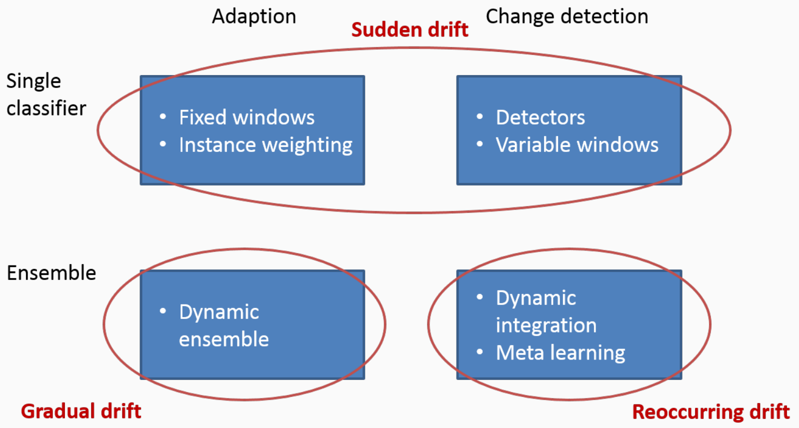


– Recurring



Handling Concept Drift

Suggested usage of techniques



Now a First, Simple, but “Real” Stream Classifier

- Among the batch classifiers, lazy learners (classifiers that do not train a model, e.g. nearest neighbor classifiers) are good candidates for direct application on streams
- Problem: upon arrival of a new object x_i, computing the kNN of x_i on the entire history can be very expansive
- The dynamic nearest neighbor[7] method maintains a fixed number of n weighted representatives as training data and uses a weighted kNN classifier on these representatives
- Online maintenance of n representatives R (when x_i arrives)
  - Determine closest representative $r \in R$
  - If $dist(x_i, r) < \tau$, add x_i to r
    - If the labels of x_i and r are equal, increase weight by 1, else decrease
    - If the resulting weight is 0 replace r by x_i in R
  - Else, add x_i to R (since n is exceeded, subtract 1 from weights of all $r \neq x_i$ and delete least weighted $r' \in R$)

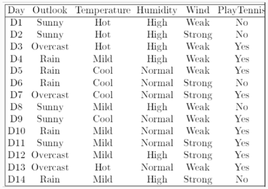
[7] Bandyopadyay et al.: On-board mining of data streams in sensor networks. Adv. Methods for K.D. from Complex Data, Springer 2005

Kapitel 3: Classification

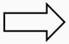
Recap

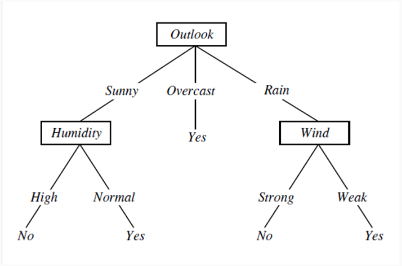
- Given a training set $D = \{(x, y)\}$ with
  - d predictive features $x = \langle x_1, \ldots, x_d \rangle$
  - class attribute $y \in \{y_1, \ldots k\}$
- Goal: find function f with $y = f(x)$
- Decision tree model
  - nodes contain tests on the predictive attributes
  - leaves contain predictions on the class attribute

Training set



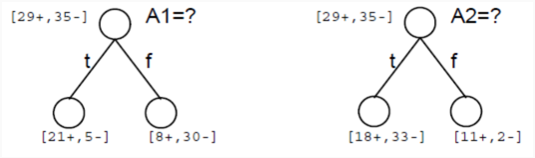
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No





Recap

- Basic algorithm (ID3, Quinlan 1986)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root node
  - Select best attribute for splitting using purity measure (Information Gain, Gini Index, ...), i.e., the one that splits the training data in the most pure partitions
  - Continue in sub-trees with the actual partition of the training set that have the corresponding feature values



Challenge

- Thus far, in order to decide on which attribute to use for splitting in a node (essential operation for building a DT), we need to have all the training set instances resulting in this node.
- But, in a data stream environment
    - The stream is infinite
    - We cant wait for ever in a node
- Can we make a valid decision based on some data?
  YES: Hoeffding Tree or Very Fast Decision Tree (VFDT) [DomingosHulten00]

Hoeffding Tree – Basic Idea

- Idea: In order to pick the best split attribute for a node, it may be sufficient to consider only a small subset of the training examples that pass through that node
  - No need to look at the whole data set (which is infinite in case of streams)
- Problem: How many instances are necessary?
- Use the Hoeffding bound!

The Hoeffding Bound

- Consider a real-valued random variable r whose range is R
  - e.g., for a probability the range is $R = 1$
  - for information gain the range is $log_2(c)$, where c is the number of classes
- Suppose we have n independent observations of r and we compute its mean $\bar{r}$
- The Hoeffding bound states that with confidence $1 - \delta$ the true mean of the variable, μ_r, is at least $\bar{r} - \varepsilon$, i.e., $P(\mu_r \geq \bar{r} - \varepsilon) = 1 - \delta$
- The error ε is given by

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

The Hoeffding Bound

- This bound holds true regardless of the distribution generating the values
- It depends only on
  - the range of values R,
  - the number of observations n, and
  - the desired confidence δ
- A disadvantage of being so general is that it is more conservative than a distribution-dependent bound

Application of the Hoeffding Bound

- Let $G()$ be the heuristic measure for choosing the split attribute at a node (e.g. Information Gain)
- After seeing n instances at this node, let
  - X_1 : be the attribute with the highest observed $G()$
  - X_2 : be the attribute with the second-highest observed $G()$
- Assumption: third-best and lower attributes have sufficiently smaller gains so that their probability of being the true best choice is negligible

Application of the Hoeffding Bound

- Let $\Delta G = |G(X_1) - G(X_2)|$ be the difference between the values of the two best attributes
- ΔG is the random variable that we will estimated using the Hoeffding bound, i.e. we want to estimate the true mean $\mu_{\Delta G}$ from the observed mean $\bar{\Delta G}$ (the difference after seen n instances)
- In this scenario, the Hoeffding bound states that $P(\mu_{\Delta G} \geq \bar{\Delta G} - \varepsilon) = 1 - \delta$ with

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

Application of the Hoeffding Bound

- For a desired δ, we can compute the corresponding Hoeffding bound ε for estimating $\mu_{\Delta G}$ by $\bar{\Delta G}$ (computed after seeing n observations at the given node)
- The Hoeffding bound guarantees that with probability $1-\delta$ we have $\mu_{\Delta G} \geq \bar{\Delta G} - \varepsilon$
- Therefore, if $\bar{\Delta G} \geq \varepsilon$, we know that the true difference $\mu_{\Delta G} > 0$ (because $\mu_{\Delta G} \geq \bar{\Delta G} - \varepsilon > 0$), thus, we can choose X_1 for splitting at this node with confidence $1-\delta$
- Otherwise, i.e., if $\bar{\Delta G} < \varepsilon$, the sample size is not enough for a stable decision because we cannot guarantee that $\mu_{\Delta G} > 0$
  - With R and δ fixed, the only variable left to change ε is n
  - We need to extend the sample by seeing more instances, until $\bar{\Delta G}$ becomes larger than the corresponding ε

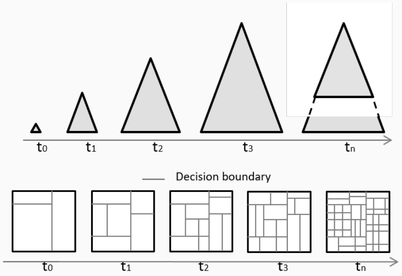
Very Fast Hoeffding Trees (VFHT)

- The VFDT[8] learner optimizes the Hoeffding tree as follows:
- Ties:
  - When ≥ 2 attributes have very similar G-value, potentially many examples will be required to decide between them with high confidence
  - This is presumably wasteful, as it makes little difference which feature is chosen
  - Break it by splitting on current best if current $\Delta G < \varepsilon < \tau$, where τ is a user-specified threshold
- G computation:
  - It is inefficient to recompute G for every example, since it is unlikely that the split decision will be made at that specific point
  - VFDT allows the user to specify a minimum number of examples that must be accumulated before G is recomputed
  - This may implement a smaller δ, but yields a speedup

[8] Domingos and Hulten.: Mining High-Speed Data Streams. KDD 2000

Limitations of HTs

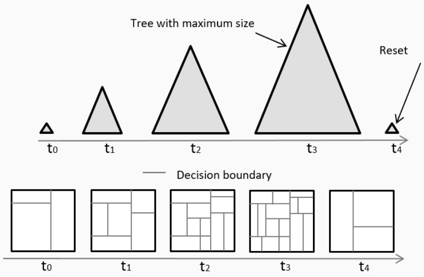
- The HT accommodates new instances from the stream
- But, does not delete anything (does not forget!)
- With time
  - The tree becomes more complex (overfitting is possible)
  - The historical data (used for finding the splits on the highest levels of the tree) dominate its decisions (difficult to adapt to changes)



HT over time [Mahmud15]

Adaptive Size HT (ASHT)

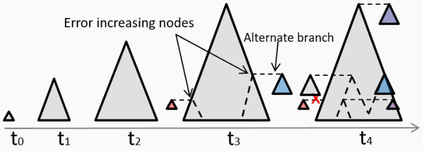
- Introduces a maximum size (# of splitting nodes) bound
- When the limit is reached, the tree is reset (test for the limit after each node split)
- Due to the reset, the tree forgets but looses all information learned so far

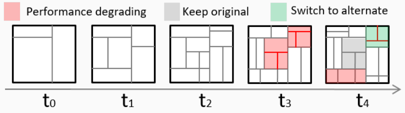


ASHT over time [Mahmud15]

Concept-adapting HT (CAHT)

- Starts maintaining an alternate sub-tree when the performance of a node decays
- When the new sub-tree starts performing better, it replaces the original one
- If original sub-tree keeps performing better, the alternate sub-tree is deleted and the original one is kept





AdaHT over time [Mahmud15]

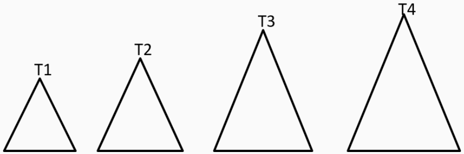
ASHT Ensembles

Idea:

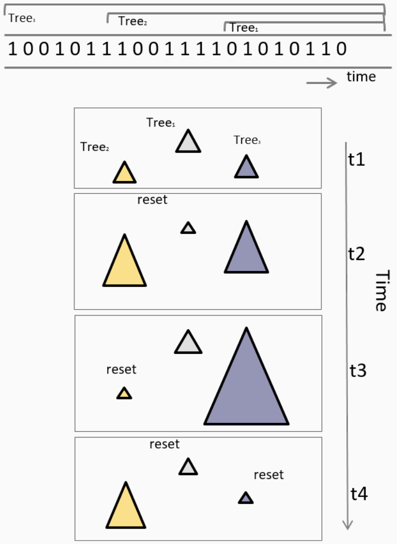
- Instead of a single model, use a combination of k models to increase accuracy
- Combine a series of k learned models, $M_1, ..., M_k$, with the aim of creating an improved consensus (ensemble) model $M*$
- To predict the class of previously unseen records, aggregate the predictions of the ensemble
- Combination methods
  - Bagging: use Bootstrapping to generate small training samples and learn one model at each sample
  - Boosting: sequentially train and test the M_i's; misclassified samples get higher weights for the remaining models
  - Stacking: use a meta learner that takes the predictions of the M_i's as input vector

ASHT Ensembles

- Bagging using ASHTs of different sizes
- Smaller trees adapt more quickly to changes
- Larger trees perform better during periods with no or little change
- The maximally allowed size for the *n*th ASHT tree is twice the maximally allowed size for the (*n*-1)th tree.
- Each tree has a weight proportional to the inverse of the square of its error
- The goal is to increase bagging performance by tree diversity



ASHT Ensembles



HT Family — Overview

- All HT, AdaHT, ASHT accommodate new instances from the stream
- Basic HT does not forget
- ASHT forgets by resetting the tree once its size reaches its limit
- AdaHT forgets by replacing sub-trees with new ones
- Bagging ASHT uses varying size trees that respond differently to change

Kapitel 3: Classification

Recap

- Given an instance x with attributes $(a_1, a_2, \ldots, a_d)$
- Predict class label $c \in C$ that maximizes the a posteriori probability $p(c|x)$, which can be computed according to Bayes' rule by the class prior $p(c)$ and the a priori likelihood $p(x|c)$:

$$c = \text{argmax}_{c \in C} p(x|c)p(c)$$

- The class prior $p(c)$ can be estimated from the training set as its relative frequency
- The Instance likelihood $p(x|c) = p(x_1 x_. .. x_d|c)$ is much harder
- One way (Naive Bayes) is to assume attribute independence:

$$p(x_1 x_. .. x_d|c) = \prod p(x_i|c) = p(x_1|c \cdot \ldots \cdot p(x_d|c)$$

and the $p(x_i|c)$ can be estimated from the training set

From Batch to Streams

- How can we maintain the model estimates over time based on the stream?
- How can we include new instances in the model?
- How can we forget obsolete instances?
- In what follows, we assume a stream of documents (text data)
- The solutions though are not limited to text

Batch Naive Bayes on Text Data

- Given a training set D of documents with vocabular V, and $C = \{+, -\}$ (positive/negative)
  - N_c is the number of documents of class c
  - N_{ic} is the number how often word w_i occurs in all documents of class c
- For a new document d we would like to predict its class $c \in C$ given the words w_i in d
- Using Naive Bayes this is done as follows

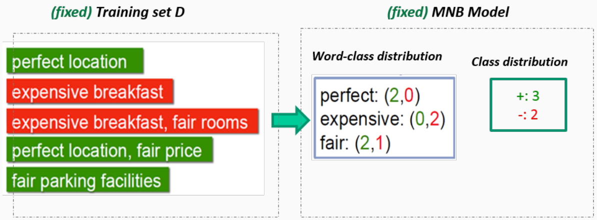
$$p(c|d) = p(c) \cdot p(d|c) = p(c) \cdot \prod_{w_i \in d} p(w_i|c)^{f_i^d}$$

where

  - $p(c) = N_c / |D|$ is the class prior
  - $p(w_i|c) = \frac{N_{ic}}{\sum_j^{|V|} N_{jc}}$ is the word class conditional estimation
  - f_i^d is the relative frequency of word $w_i \in d$

Batch Naive Bayes on Text Data

- Visually:



$$p(c|d) = p(c) \cdot p(d|c) = p(c) \cdot \prod_{w_i \in d} p(w_i|c)^{f_i^d}$$

$$= \frac{N_c}{|D|} \cdot \prod_{w_i \in d} \left(\frac{N_{ic}}{\sum_j^{|V|} N_{jc}}\right)^{f_i^d}$$

Accumulative Naive Bayes on Text Streams

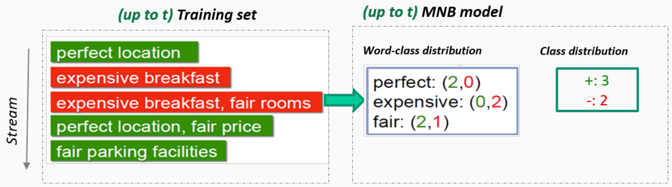
- Prediction is based on model counts up to time slot t

$$\begin{aligned} p^t(c|d) &= p(c)^t \cdot p^t(d|c) = p(c)^t \cdot \prod_{w_i \in d} p^t(w_i|c)^{f_i^d} \\ &= \frac{N_c^t}{|D|} \cdot \prod_{w_i \in d} \left(\frac{N_{ic}^t}{\sum_j^{|V|} N_{jc}^t}\right)^{f_i^d} \end{aligned}$$

  where N_c^t and N_{ic}^t are the accumulated counts from the beginning of the stream until time slot t
- Model update: add information on new d in the model which affects N_c^t and N_{ic}^t (for all words w_i occurring in d)
- Problem: Nothing is forgotten, new instances are always accumulated; difficult to adapt in times of change

Accumulative Naive Bayes on Text Streams

- Visually:



$$p^t(c|d) = p(c)^t \cdot p^t(d|c) = p(c)^t \cdot \prod_{w_i \in d} p^t(w_i|c)^{f_i^d}$$

$$= \frac{N_c^t}{|D^t|} \cdot \prod_{w_i \in d} \left(\frac{N_{ic}^t}{\sum_j^{|V|} N_{jc}^t}\right)^{f_i^d}$$

Aging-based Naive Bayes on Text Streams

- A temporal model that keeps track of the last time that an observation is made in the stream
  - For classes record the time of the last class observation t_{lo}^{c} in the stream in addition to N_c^t
  - For word-class pairs *ic* record the time of the last word-class observation t_{lo}^{ic} in the stream in addition to N_{ic}^t
- Timestamp propagation from documents to classes and word-class pairs is a temporal de-coupling of words from documents (observation updates might come from different documents)
- Allows to differentiate observations based on their recency

Aging-based Naive Bayes on Text Streams

- Incorporating exponential aging of words w_i at the current time t relative to the last occurence of w_i at t_i (and a decay rate λ):

$$age(w_i, t) = e^{-\lambda(t-t_i)}$$

- Updated temporal probability estimates

$$p^t(c) = \frac{N_c^t \cdot e^{-\lambda(t-t_{lo}^c)}}{|D^t|}$$

and

$$p^t(w_i|c) = \frac{N_{ic}^t \cdot e^{-\lambda(t-t_{lo}^{ic})}}{\sum_j^{|V|} N_{jc}^t \cdot e^{-\lambda(t-t_{lo}^{jc})}}$$

Summary: Naive Bayes

- Easy model maintenance when adding a new document *d* (update the model counts based on *d* and set *lo* in the affected entries to *t*
- Thus, Naive Bayes classifiers are ideal choices for streams
  - Popular, simple, powerful
  - allows for the seamless adaptation of the model based on new instances
  - deals with dynamic feature spaces
- Accumulative NB counts for new instances but does not forget (difficult to adapt to changes)
- Ageing-based NBs provide a temporal model that allows for ageing of the model based on the recency of the observations

Kapitel 3: Classification

Recap

- The quality of a batch classifier is evaluated over a separate test set of labeled instances
- For each test instance, its true class label is compared to its predicted class label
- Confusion matrix: A useful tool for analyzing how well a classifier performs
  - For an m-class problem, the matrix is of size $m \times m$
  - An example of a matrix for a 2-class problem

Predicted class

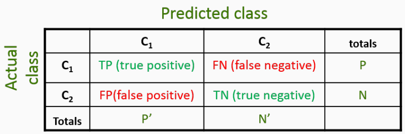
Actual class

	C_1	C_2	totals
C_1	TP (true positive)	FN (false negative)	P
C_2	FP(false positive)	TN (true negative)	N
Totals	P’	N’	

- Terminology
  - Positive tuples: tuples of the main class of interest
  - Negative tuples: all other tuples

Recap

Some (batch) evaluation measures



Predicted class

Actual class

	C_1	C_2	totals
C_1	TP (true positive)	FN (false negative)	P
C_2	FP(false positive)	TN (true negative)	N
Totals	P’	N’	

- Accuracy: % of test set instances correctly classified

$$Accuracy = \frac{TP + TN}{P + N}$$

- Error rate: % of test set instances incorrectly classified

$$Error = \frac{FP + FN}{P + N} = 1 - Accuracy$$

Recap

Some (batch) evaluation procedures

- Holdout method: data is randomly partitioned into two independent sets: Training set (approx. 2/3) for model construction, test set (approx. 1/3) for evaluation
  Problematic for small data sets (less data for training)
- Cross-validation (k-fold cross validation, usually $k = 10$):
  - Randomly partition the data into k mutually exclusive subsets $D_1, \ldots, D_k$ each approximately equal size
  - Training and testing is performed k times
  - At the i-th iteration, use D_i as test set and the rest $k - 1$ partitions as training set
  - Accuracy is the avg accuracy over all iterations
  - Thus, all available observations are exploited for training and testing
- Check out KDD I for more and a detailed discussion

Evaluation of Stream Classifiers

- Holdout method
  - 2 separate data sets for training (approx. 70% - 80% of the data set) and testing (the rest)
  - Online model training on the training set
  - Online model testing on test set
  - But static test set!!!
- Prequential evaluation (or, interleaved test-then-train evaluation)
  - One data set for training and testing
  - Models are first tested then trained in each instance
  - Test set is dynamic!!!
  - But it assumes the direct availability of the labels of the arriving instances for testing (or a separate “virtual stream” for for the training set)