Data Streams - Lecture 2

Unsupervised Learning for datastreams: Online Learning and large data sets

Dr. Jérémie Sublime – jeremie.sublime@isep.fr

ISEP - LISITE Laboratory - DASSIP Team
LIPN - CNRS UMR 7030

jeremie.sublime@isep.fr
https://sites.google.com/view/jsublime/

Schedule

- January 16th : Tools for Datastream Processing (Lecture – R. Chiky)
- January 23rd : 1st Lab on tools for Datastream Processing (R. Chiky)
- ~~January 30th : Introduction to Unsupervised Learning for datastreams (Lecture – J. Sublime)~~ (Snow)
- ~~February 6th : 2nd Lab on online and datastream Learning (J. Sublime)~~ (Homeworks)
- February 6th : **Introduction to Unsupervised Learning for datastreams (Lecture – J. Sublime)**
- February 13th : Datastream Processing and Analytics (Lecture – A. Bondu)
- February 20th : Lab on Datastream Processing (A. Bondu)
- February 27th : Article presentations (You guys – J. Sublime)

Lecture Outline

1. Revisions
2. Online Clustering and Datastreams
3. Example : K-Means algorithm adaptation to datastreams
4. Conclusions

Outline

From previous classes

In the previous course

- You saw basic definitions of what a datastream is, and the underlying basic issues and concepts
- You were introduced to some high level tools and software to deal with datastreams

From previous classes

In the previous course

- You saw basic definitions of what a datastream is, and the underlying basic issues and concepts
- You were introduced to some high level tools and software to deal with datastreams

In this course

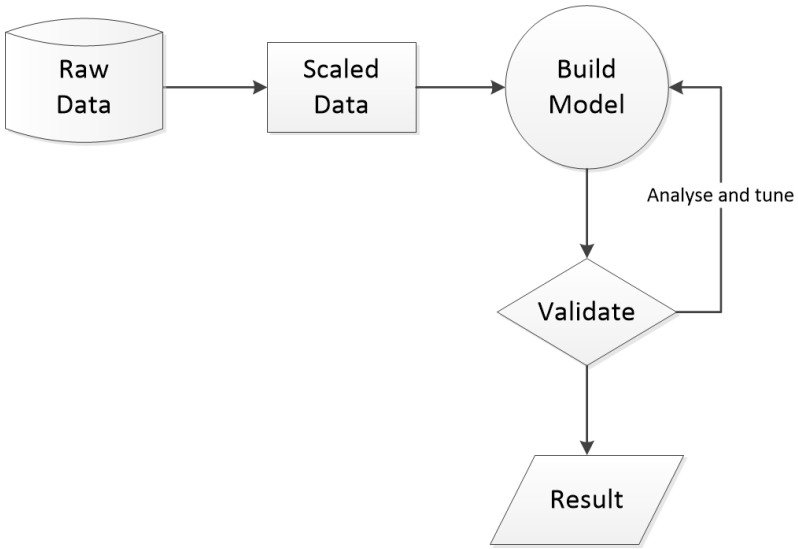
- We are going to talk about lower level Machine Learning aspects of datastream processing
- The focus is going to be on unsupervised learning applied to datastreams

Unsupervised Learning : some reminders

- Learning from unlabeled data
- An exploratory task : Finding structures (clusters/partitions), visualizing (dimension reduction), etc.

Unsupervised Learning : some reminders

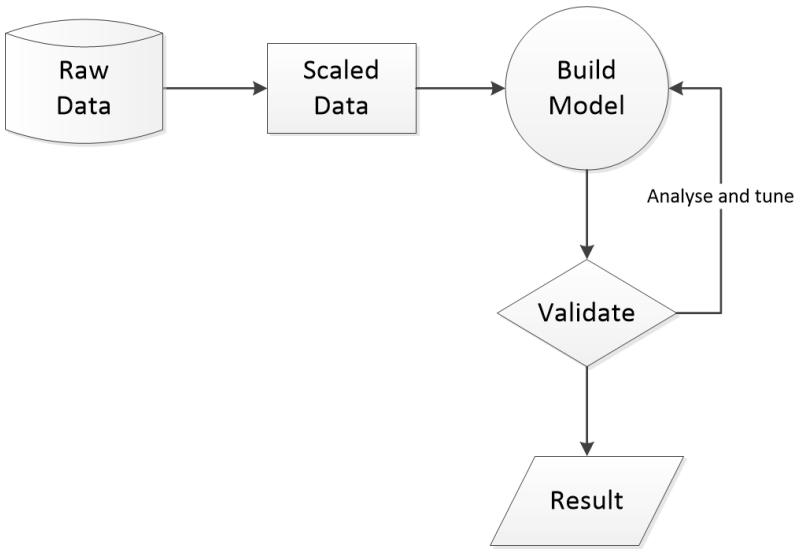
- Learning from unlabeled data
- An exploratory task : Finding structures (clusters/partitions), visualizing (dimension reduction), etc.



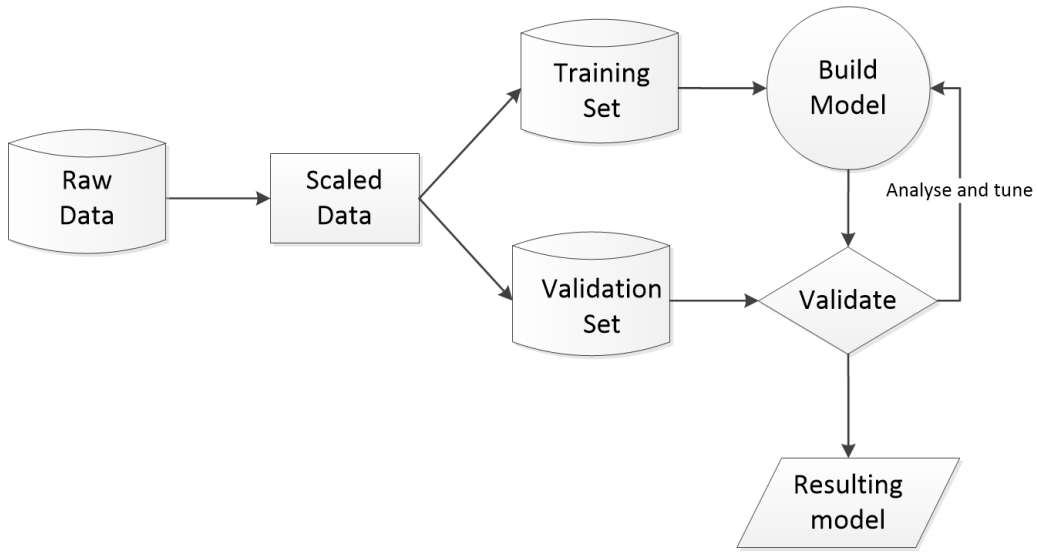
(c) Unsupervised learning flow

Unsupervised Learning : some reminders

- Learning from unlabeled data
- An exploratory task : Finding structures (clusters/partitions), visualizing (dimension reduction), etc.

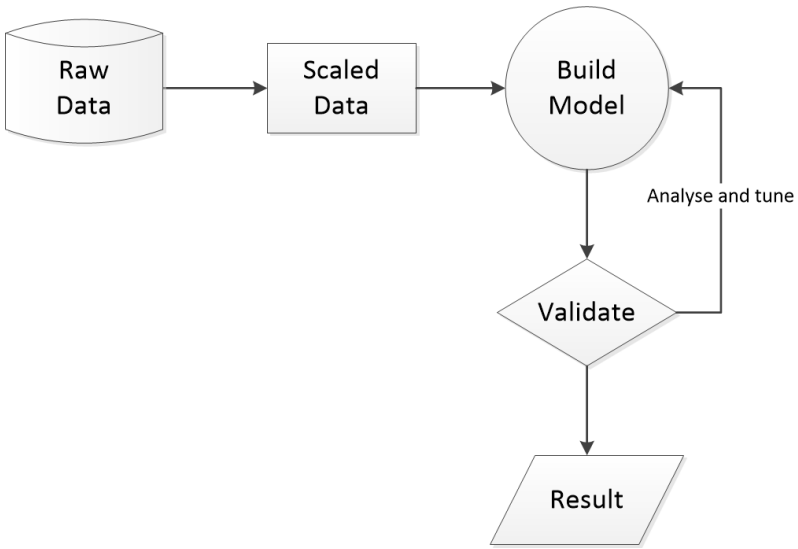


(e) Unsupervised learning flow

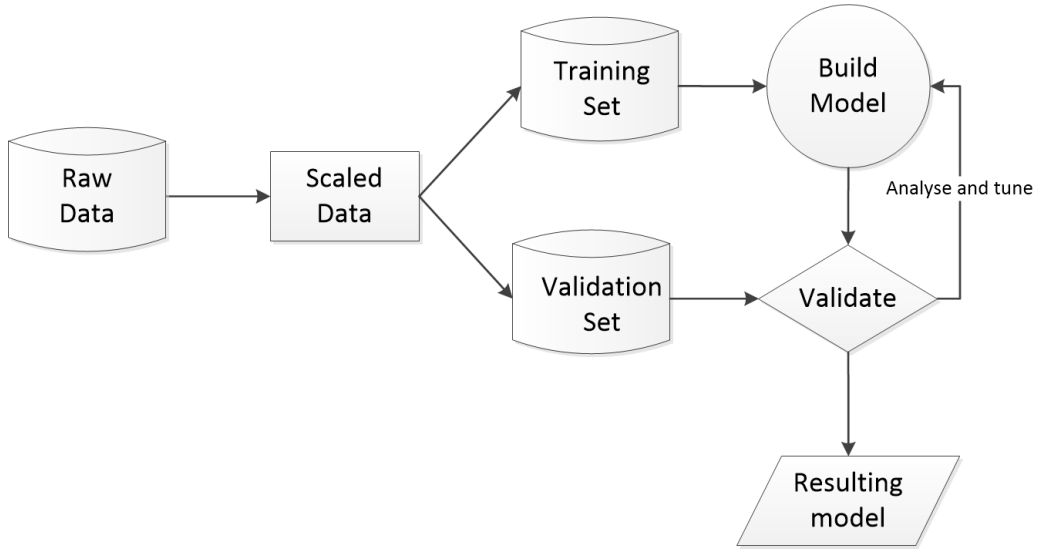


(f) Supervised learning flow

Why is unsupervised learning intuitively more problematic for datastreams ?

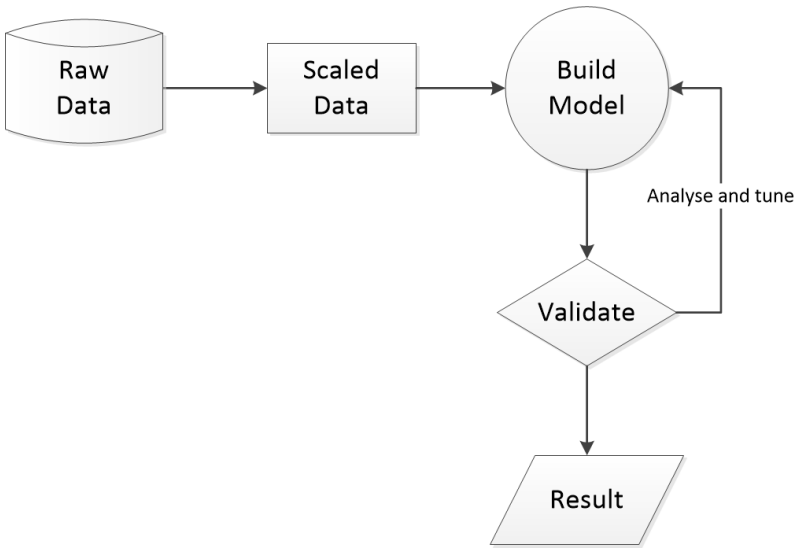


(g) Unsupervised learning flow

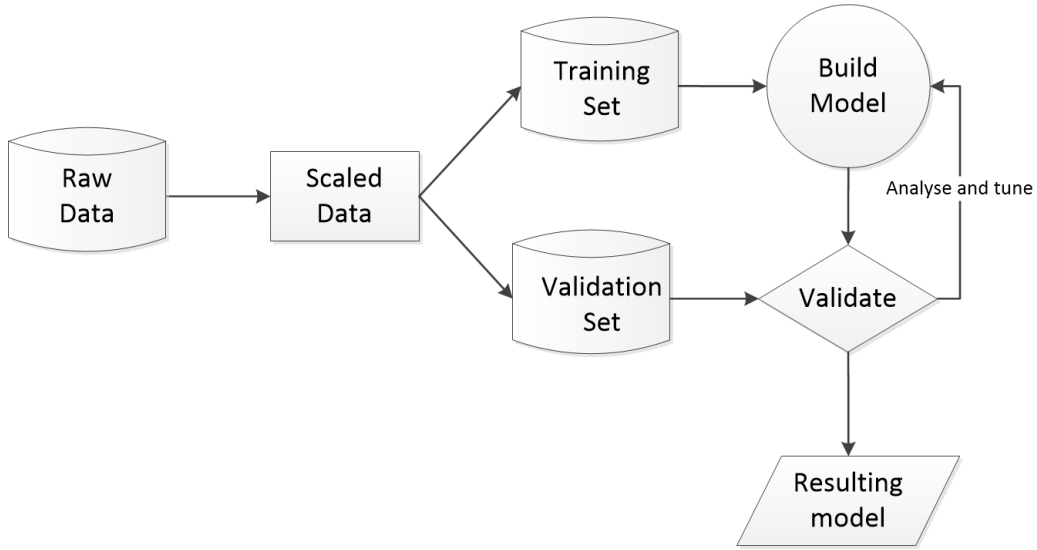


(h) Supervised learning flow

Why is unsupervised learning intuitively more problematic for datastreams ?



(i) Unsupervised learning flow



(j) Supervised learning flow

- In supervised learning, you can train and validate on static data and then classify data from streams on the fly without much issues.
- In unsupervised learning everything has to be done on the stream itself and this can be a problem.

What is clustering ?

Clustering : Definition

Clustering is an **unsupervised learning** task whose aim is to regroup data so that :

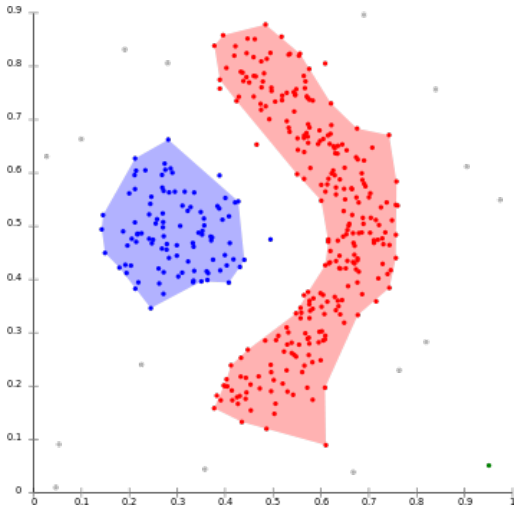
- Data that are *similar* end up in the same group called **cluster**
- Data that are *dissimilar* are in different clusters

- A **cluster** is a group of similar data based on a **predefined criterion**
- Data are typically reprensented as :
  - Vector from an Euclidian space
  - As a pairwise distance matrix (Jaccard, cosine, etc.) in a metric space
- **Anomaly detection** (outlier detection) is a common clustering task

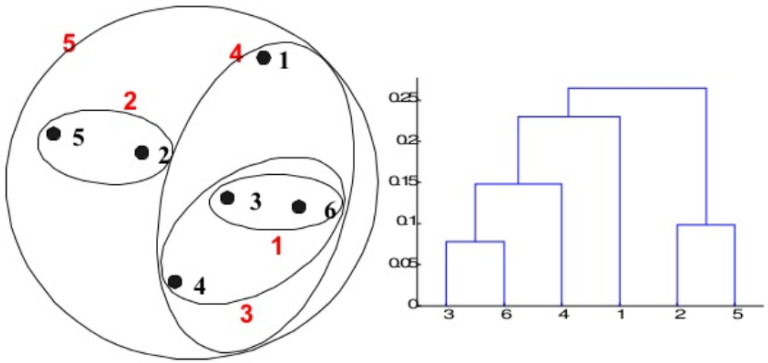
Examples of clustering tasks

- Regrouping customers based on common features or habits
- Sorting texts or tweets based on keywords
- Regrouping DNA sequence based on their distance of edition
- Detecting frauds and anomalies
- ...

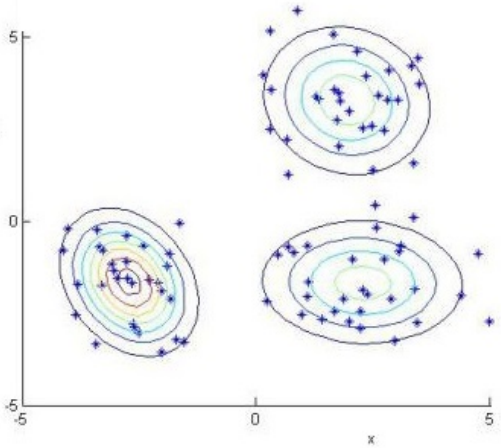
Different families for clustering algorithms



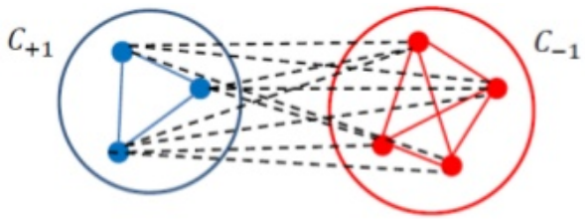
(k) Density based methods



(l) Hierarchical methods



(m) Model/prototype based methods



(n) Partition based methods

Open problems and challenges

Examples of open problems :

- Picking a similarity measure
- Picking a clustering algorithm and the rights parameters to properly detects structures that are unknown in advance
- Evaluating and comparing clustering partitions

Open problems and challenges

Examples of open problems :

- Picking a similarity measure
- Picking a clustering algorithm and the rights parameters to properly detects structures that are unknown in advance
- Evaluating and comparing clustering partitions

Challenges for today's class

Most clustering algorithms were designed for "small" datasets with little to no memory or time constraint. These algorithms do not work well with modern datasets :

- How to proceed when the data are too big to fit in memory ?
- How to handle continuous data flows and data that must be processed on the fly ?
- How to deal with non-stationary environments where clusters can evolve through time ?

Outline

Definitions

Batch, incremental, Online, and datastreams learning, what differences ?

Definitions : Batch vs Incremental

Batch learning

- Learning is done on the whole dataset
- The data can fit in memory and can be browsed as many time as needed
- Stationary structures (clusters)
- No "strong time constraint" to learn

Definitions : Batch vs Incremental

Batch learning

- Learning is done on the whole dataset
- The data can fit in memory and can be browsed as many time as needed
- Stationary structures (clusters)
- No "strong time constraint" to learn

Incremental learning

- Learning is done incrementally **as data are received**
- All or part of the data can be kept in memory once they have been received
- The clusters **might be non-stationary**
- Mild time constraints to learn : the model needs to be updated as new data arrive

Definitions : Online Learning

Online Learning

- Learning is done incrementally **as data are received**
- Data can't be stored in memory, **each example is processed only once**
- The clusters **may be non-stationary**
- Learning must be done **in real time** for each data

Definitions : Online Learning

Online Learning

- Learning is done incrementally **as data are received**
- Data can't be stored in memory, **each example is processed only once**
- The clusters **may be non-stationary**
- Learning must be done **in real time** for each data

Remark : The difference between online and incremental learning is not always clear in the literature. The term "Online" learning tends to be predominant since most real life application are Online.

Definitions : Time series

Time series

- Learning is done on data that have **time dependencies** (not iid)
- Depending on the application, it may be batch, incremental or online learning
- The structures **may be non-stationary**
- May or may not be real time depending on the application

Definitions : Time series

Time series

- Learning is done on data that have **time dependencies** (not iid)
- Depending on the application, it may be batch, incremental or online learning
- The structures **may be non-stationary**
- May or may not be real time depending on the application

Remark : Many methods linked to time series analysis are not real time : ARIMA, HMM.

Definitions : datastreams

Learning from datastreams

- Learning done on **data that cannot be fully loaded in memory** and will be processed as **streams**.
- Quite often, there will be **time dependencies**.
- Data can be kept temporarily in memory, but **each example is processed only once**
- Structures may or may not be stationary.
- May or may not be real time depending on the application

Definitions : datastreams

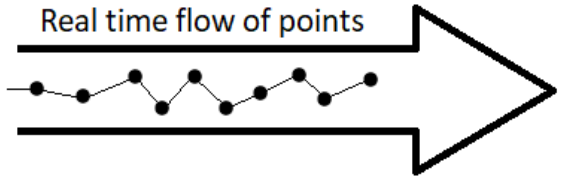
Learning from datastreams

- Learning done on **data that cannot be fully loaded in memory** and will be processed as **streams**.
- Quite often, there will be **time dependencies**.
- Data can be kept temporarily in memory, but **each example is processed only once**
- Structures may or may not be stationary.
- May or may not be real time depending on the application

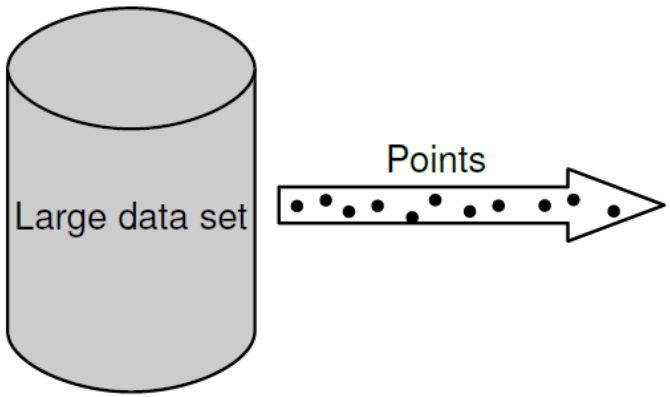
Remark : Learning from large datasets de facto shares many common characteristics with both online and time series learning (even thought datastreams don't always have time dependencies).

Datastreams : 2 cases

- "Regular datastream" : Real time flow of data with time dependencies (can't start over ! Infinite stream ?)



- "Artificial datastream" : Dataset too big to fit in memory : the data arrive one by one or by groups using sampling techniques (can't start over either ! Finite stream.)



Datastreams : 2 cases with examples

Real streams

- Energy consumption data
- Traffic data
- Network traffic data
- Monitoring data (e.g. medical, physics, etc.)
- etc.

Artificial streams

- Genetic data
- Web data
- large video or teledetection data
- Basically anything that you cannot fully load in RAM.

Learning from datastreams with time dependencies

There are a few possibilities to learn from datastreams that have time dependencies :

- Using **Online learning** algorithms to learn in real time as examples are loaded one by one in memory.
- Learning on **time windows**.

Learning from datastreams with time dependencies

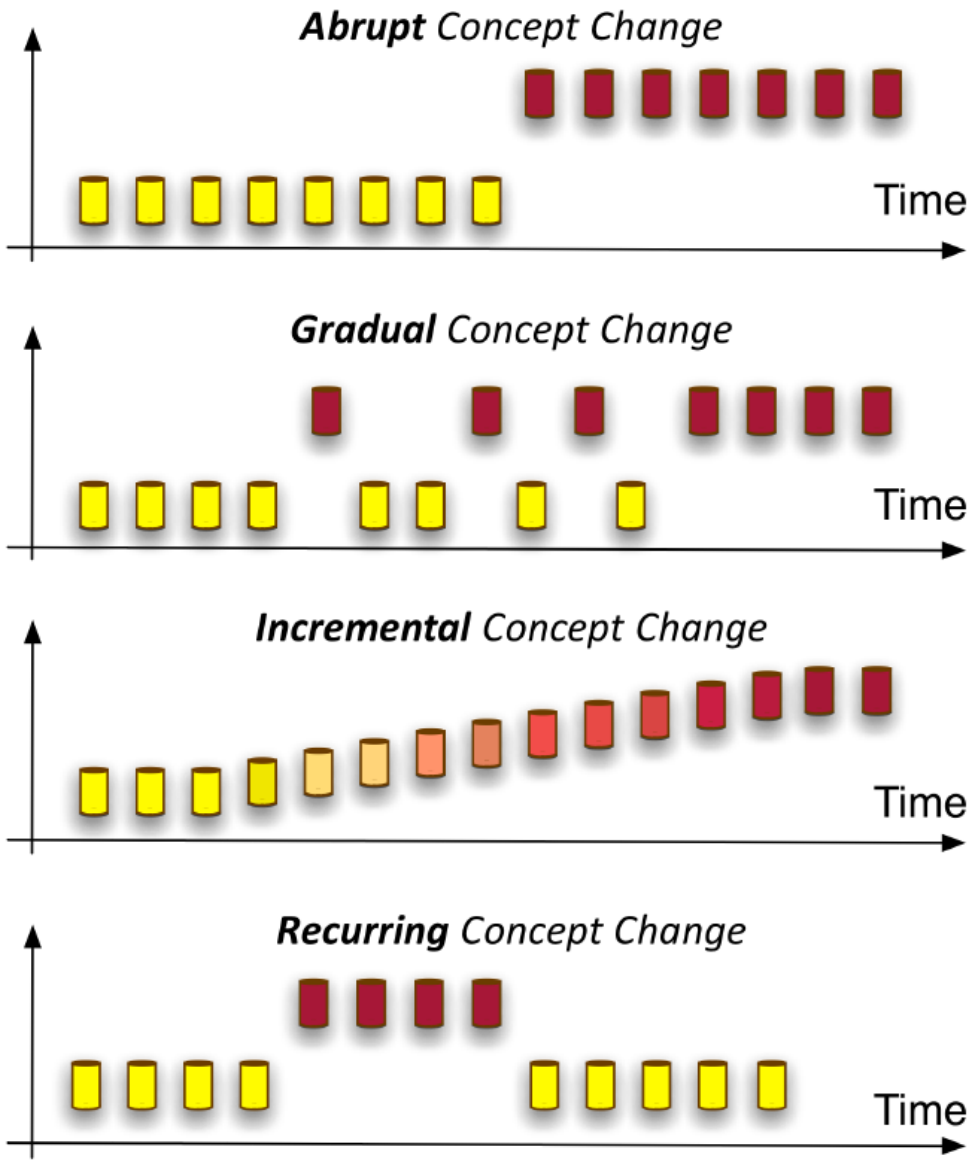
There are a few possibilities to learn from datastreams that have time dependencies :

- Using **Online learning** algorithms to learn in real time as examples are loaded one by one in memory.
- Learning on **time windows**.

Challenges

- Detect change from noise
- Handle concept drift
- Learn in real time
- Use the time dependencies ? (When ? How ?)
- Data normalization is impossible : careful with distance functions !

Concept drift in non-stationary environments



Desirable properties to handle concept drift

- Quick detection and adaptation to change
- Distinguishing noise from change
- Recognizing and handling recurrent patterns
- Can learn and adapt with limited resources (time and memory)

Concept drift

Issues

- Training an algorithm as accurate as possible
  - Costs time and memory
- Having an algorithm that adapts quickly and resits to noise
  - Quickly forget

Concept drift

Issues

- Training an algorithm as accurate as possible
  - Costs time and memory
- Having an algorithm that adapts quickly and resits to noise
  - Quickly forget

One possible solution : Learning on a "sliding time window"

- The model is built and updated based on a smaller or larger time window of data, rather than based on all previously seen data.
- It is a desirable improvement from regular incremental learning.

Concept drift and windowing (1/2)

Concept drift and windowing (2/2)

A key issue with windowing techniques is to pick the right size for the sliding window :

Concept drift and windowing (2/2)

A key issue with windowing techniques is to pick the right size for the sliding window :

Smaller window

- Adapts very fast
- Lesser accuracy

Concept drift and windowing (2/2)

A key issue with windowing techniques is to pick the right size for the sliding window :

Smaller window

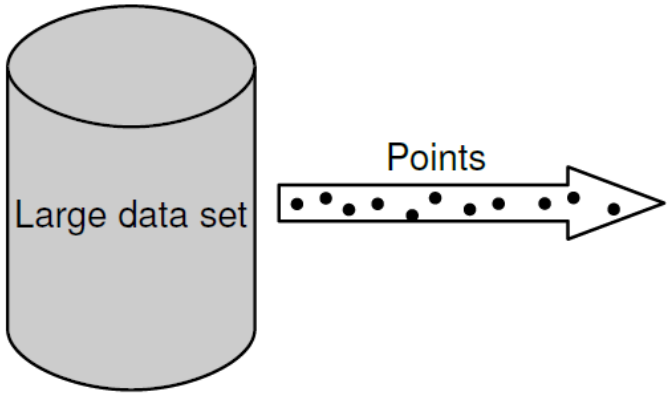
- Adapts very fast
- Lesser accuracy

Larger window

- Low adaptability
- Stable and very precise model (good for stationary environments)

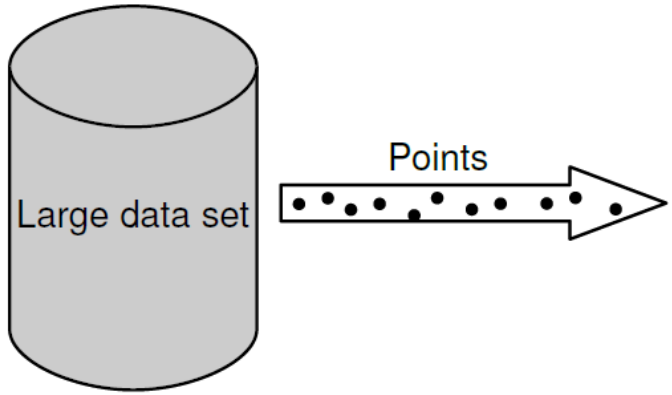
Learning from artificial data streams

- "Artificial datastream" : Dataset too big to fit in memory : the data arrive one by one or by groups (can't start over either.)



Learning from artificial data streams

- "Artificial datastream" : Dataset too big to fit in memory : the data arrive one by one or by groups (can't start over either.)



Challenges

- Detect small clusters from noise
- Make a global analysis from samples
- Compress the data ?
- Learn in real time ?
- Data normalization is difficult : careful with distance functions !

Learning in real time from artificial datastreams

Learning from artificial datastreams in real time is pretty much the same than learning from regural datastreams in a stationary environment :

- Online and incremental learning can still be used.
- No need to worry about concept drift and time dependencies.

Learning in real time from artificial datastreams

Learning from artificial datastreams in real time is pretty much the same than learning from regural datastreams in a stationary environment :

- Online and incremental learning can still be used.
- No need to worry about concept drift and time dependencies.

Tools for real time artificial datastreams

Use regular datastreams algorithms : Very high rate of success.

When real time is not needed

When real time is not needed, much simpler techniques may be used :

- Use the stream to build a summarized/compressed version of the original data and applying regular clustering to it : get a global view of a dataset processed as a stream.
- Learn in parallel on subsets of the data and merge the results

When real time is not needed

When real time is not needed, much simpler techniques may be used :

- Use the stream to build a summarized/compressed version of the original data and applying regular clustering to it : get a global view of a dataset processed as a stream.
- Learn in parallel on subsets of the data and merge the results

Issues

- Properly identifying small clusters in subsets (possible confusion with noise)
- Losing too much information during the compression process
- All sort of sampling problems can happen

Regular VS Artificial datastreams

Regular streams : Challenges

- Detect cluster change from noise
- Concept drift
- Learn in real time
- Time dependencies ?
- Data normalization is problematic

Artificial streams : Challenges

- Detect small clusters from noise
- Global analysis from samples
- Learn in real time ?
- Compression techniques ?
- Data normalization is problematic

Regular streams : Solutions

- Online learning
- Windowing techniques with online learning

Artificial streams : Solution

- Online learning
- Sampling and compression techniques + regular clustering

Outline

K-Means : Principle

- Assume all data are in Euclidan space : $x_i \in \mathbb{R}^d$
- Clusters are represented using centroids $\mu_j \in \mathbb{R}^d$
- Each data is assigned to the closest centroid

Objective function

Goal : Finding the centroids μ that minimize :

$$L(\mu) = \sum_{i=1}^{N} \min_j ||\mu_j - x_i||_2^2$$

K-Means : Principle

- Assume all data are in Euclidan space : $x_i \in \mathbb{R}^d$
- Clusters are represented using centroids $\mu_j \in \mathbb{R}^d$
- Each data is assigned to the closest centroid

Objective function

Goal : Finding the centroids μ that minimize :

$$L(\mu) = \sum_{i=1}^{N} \min_j ||\mu_j - x_i||_2^2$$

- Non-convex optimization !
- **NP-hard** : Can't solve optimally in general

K-Means : Algorithm

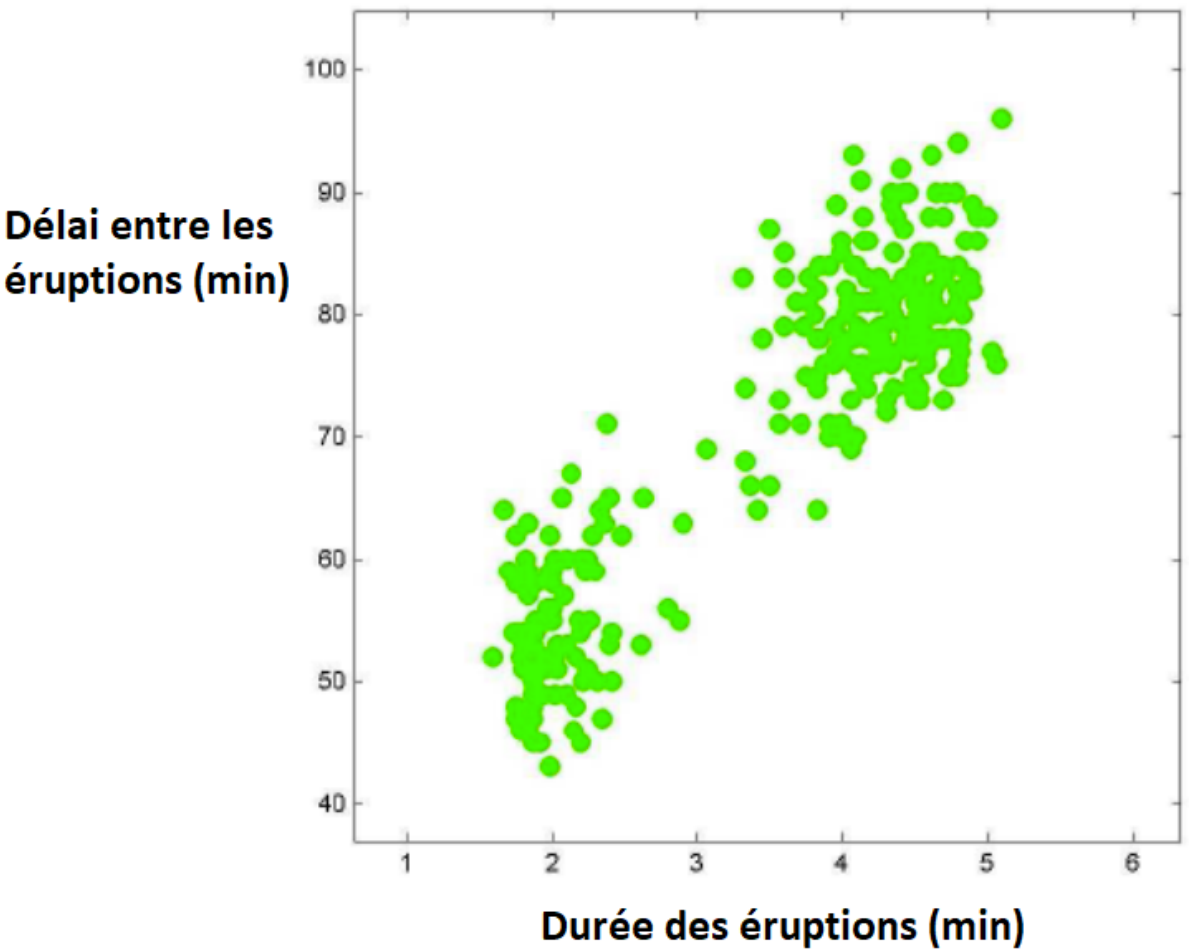
- Initialize the K cluster centers (usually pick one at random and pick the other ones with maximum distance)
- While the centroids are changing :
  - Assign each data to the closest centroid :

$$c_i = \underset{j}{\text{argmin}} \, ||\mu_j - x_i||_2^2$$

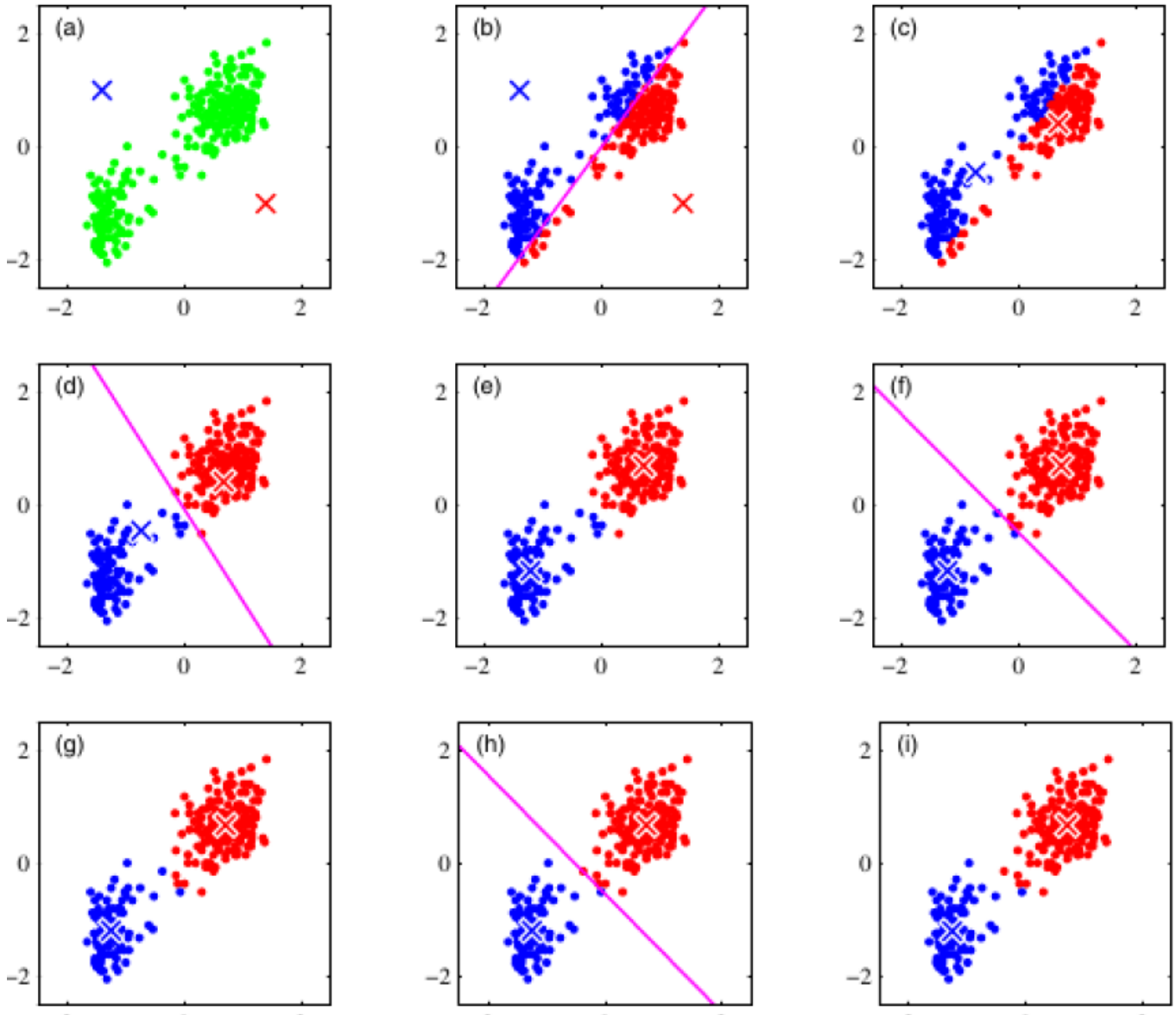
  - Update the centroids :

$$\mu_j = \frac{1}{n_j} \sum_{i|c_i=j} x_i$$

K-Means : Example (1/2)



K-Means : Example (2/2)



K-Means : Properties

- The objective function is guaranteed to **monotonically decrease**

$$L(\mu)^{(t)} = \sum_{i=1}^{N} \min_{j} ||\mu_j - x_i||_2^2$$

$$L(\mu)^{(t+1)} \leq L(\mu)^{(t)}$$

K-Means : Properties

- The objective function is guaranteed to **monotonically decrease**

$$L(\mu)^{(t)} = \sum_{i=1}^{N} \min_{j} ||\mu_j - x_i||_2^2$$

$$L(\mu)^{(t+1)} \leq L(\mu)^{(t)}$$

- Convergence toward a local optimum

K-Means : Properties

- The objective function is guaranteed to **monotonically decrease**

$$L(\mu)^{(t)} = \sum_{i=1}^{N} \min_{j} ||\mu_j - x_i||_2^2$$

$$L(\mu)^{(t+1)} \leq L(\mu)^{(t)}$$

- Convergence toward a local optimum
- Complexity per iteration : $O(n \cdot d \cdot K)$

K-Means : Properties

- The objective function is guaranteed to **monotonically decrease**

$$L(\mu)^{(t)} = \sum_{i=1}^{N} \min_j ||\mu_j - x_i||_2^2$$

$$L(\mu)^{(t+1)} \leq L(\mu)^{(t)}$$

- Convergence toward a local optimum
- Complexity per iteration : $O(n \cdot d \cdot K)$
- Requires to process **all** the data at each iteration

K-Means for datastreams

Ideas

- Computing the means requires **all data** at **each iteration** : that's a problem

K-Means for datastreams

Ideas

- Computing the means requires **all data** at **each iteration** : that's a problem
- Mean values can actually be computed incrementally.

K-Means for datastreams

Ideas

- Computing the means requires **all data** at **each iteration** : that's a problem
- Mean values can actually be computed incrementally.
- Having to do it at each iteration still is a problem

K-Means for datastreams

Ideas

- Computing the means requires **all data** at **each iteration** : that's a problem
- Mean values can actually be computed incrementally.
- Having to do it at each iteration still is a problem

Question : Is it possible to get good results with only one pass on the data ?

Computing a mean value on the fly

- Regular Formula :

$$\mu_c = \frac{1}{n_c} \sum_{i|c_i=c} x_i$$

- Incremental formula : adding a data x_t at time t :

$$\mu_c^{(t)} = \frac{t-1}{t}\mu_c^{(t-1)} + \frac{1}{t}x_t = \mu_c^{(t-1)} + \frac{1}{t}(x_t - \mu_c^{(t-1)})$$

Computing a mean value on the fly

- Regular Formula :

$$\mu_c = \frac{1}{n_c} \sum_{i|c_i=c} x_i$$

- Incremental formula : adding a data x_t at time t :

$$\mu_c^{(t)} = \frac{t-1}{t}\mu_c^{(t-1)} + \frac{1}{t}x_t = \mu_c^{(t-1)} + \frac{1}{t}(x_t - \mu_c^{(t-1)})$$

We can use this Equation to build an "online" K-Means, where data arrive one by one.

Online K-Means : Algorithm

- Randomly initialize the K initial centers
- While data keep coming : Pour $t = 1 : T$
  - Assign the new data to the nearest center :

$$c = \underset{j}{\text{argmin}} \, ||\mu_j - x_t||_2^2$$

  - Update this center :

$$\mu_c = \mu_c + \eta_t(x_t - \mu_c)$$

Online K-Means : Algorithm

- Randomly initialize the K initial centers
- While data keep coming : Pour $t = 1 : T$
  - Assign the new data to the nearest center :

$$c = \underset{j}{\text{argmin}} \, ||\mu_j - x_t||_2^2$$

  - Update this center :

$$\mu_c = \mu_c + \eta_t(x_t - \mu_c)$$

Questions :

- How do we normalize data on the fly to avoid issues with the Euclidian distance ?
- Does it converge ?
- Under which conditions ?

Online K-Means : The data normalization issue

Reminder on data normalization and scaling

In regular Machine Learning, data normalization and scales is used to avoid issues with Euclidian like distances : attributes with larges values would otherwise crush the ones on smaller scales and the distance function would be very biased.

What about the Online k-Means and Datastream clustering in general ?

Online K-Means : The data normalization issue

Reminder on data normalization and scaling

In regular Machine Learning, data normalization and scales is used to avoid issues with Euclidian like distances : attributes with larges values would otherwise crush the ones on smaller scales and the distance function would be very biased.

What about the Online k-Means and Datastream clustering in general ?

- Without access to all the data and sometimes no memory, regular normalization and scaling is impossible in datastream learning.

Online K-Means : The data normalization issue

Reminder on data normalization and scaling

In regular Machine Learning, data normalization and scales is used to avoid issues with Euclidian like distances : attributes with larges values would otherwise crush the ones on smaller scales and the distance function would be very biased.

What about the Online k-Means and Datastream clustering in general ?

- Without access to all the data and sometimes no memory, regular normalization and scaling is impossible in datastream learning.
- Some solutions :
  - The artisanal one : Use the "Field Knowledge" to manually scale some of the attributes.

Online K-Means : The data normalization issue

Reminder on data normalization and scaling

In regular Machine Learning, data normalization and scales is used to avoid issues with Euclidian like distances : attributes with larges values would otherwise crush the ones on smaller scales and the distance function would be very biased.

What about the Online k-Means and Datastream clustering in general ?

- Without access to all the data and sometimes no memory, regular normalization and scaling is impossible in datastream learning.
- Some solutions :
  - The artisanal one : Use the "Field Knowledge" to manually scale some of the attributes.
  - The data scientist one : Use a smart distance function that learns how to weight the attributes.

Online K-Means : Convergence

To converge toward a local optimum, we need :

$\mu_c = \mu_c + \eta_t(x_t - \mu_c)$ with $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$

Case 1 : No concept drift

- A trivial solution consists in keeping all previous information in the model : $\eta_t = \frac{1}{t}$

Online K-Means : Convergence

To converge toward a local optimum, we need :

$\mu_c = \mu_c + \eta_t(x_t - \mu_c)$ with $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$

Case 1 : No concept drift

- A trivial solution consists in keeping all previous information in the model : $\eta_t = \frac{1}{t}$

Case 2 : With concept drift

- Solution using a time window of size w that forgets old examples
- Example : $\eta_t = \frac{1}{min(t,w)}$ gives more weight to new points. We have our windowing technique ! (remark : the convergence is lost).

Online K-Means : Convergence

To converge toward a local optimum, we need :

$\mu_c = \mu_c + \eta_t(x_t - \mu_c)$ with $\sum_t \eta_t = \infty$ and $\sum_t \eta_t^2 < \infty$

Case 1 : No concept drift

- A trivial solution consists in keeping all previous information in the model : $\eta_t = \frac{1}{t}$

Case 2 : With concept drift

- Solution using a time window of size w that forgets old examples
- Example : $\eta_t = \frac{1}{min(t,w)}$ gives more weight to new points. We have our windowing technique ! (remark : the convergence is lost).

- Using the right value for η_t, the Online K-Means algorithm can therefore be applied to all kinds of datastreams regardless of whether or not there are time dependencies and concept drift issues.

Online K-Means : Limits

- Sensitive to the order of the data
- K still need to be known in advance
  - Clusters appearing and disappearing is not handled well
- K-Means is dependent on its initialization : Impossible to try several initial configurations here (only one path)
- Cold start problem
- Unclear how to parallelize

Ideas on improving Online K-Means

A few approaches exist to tackle to problem of appearing and disappearing clusters in online and incremental K-Means :

- Use the X-Mean algorithm : start with 2 clusters and add extra ones when needed based on a Bayesian criterion
  - Does not handle cluster removal
  - Sensitive to abrupt changes

Ideas on improving Online K-Means

A few approaches exist to tackle to problem of appearing and disappearing clusters in online and incremental K-Means :

- Use the X-Mean algorithm : start with 2 clusters and add extra ones when needed based on a Bayesian criterion
  - Does not handle cluster removal
  - Sensitive to abrupt changes
- Use thresholds to create clusters when a data is too far from existing clusters, and to remove clusters when no data has been assigned to a cluster for a while.
  - Difficult to pick the right thresholds.
  - Sensitive to recurring changes

Ideas on improving Online K-Means

A few approaches exist to tackle to problem of appearing and disappearing clusters in online and incremental K-Means :

- Use the X-Mean algorithm : start with 2 clusters and add extra ones when needed based on a Bayesian criterion
  - Does not handle cluster removal
  - Sensitive to abrupt changes
- Use thresholds to create clusters when a data is too far from existing clusters, and to remove clusters when no data has been assigned to a cluster for a while.
  - Difficult to pick the right thresholds.
  - Sensitive to recurring changes
- Use the first data as centroids to alleviate the cold start problem.
  - Parameter sensitive
  - Data order sensitive

K-Means for large datasets

Problem

- Learn a global model from a dataset that does not fit in memory and arrives as a stream.

K-Means for large datasets

Problem

- Learn a global model from a dataset that does not fit in memory and arrives as a stream.

Idea

- Efficiently build a compact version $\mathcal{C}$ of the data $\mathcal{D}$ using sampling (can be handled with streams) such that K-Means results on $\mathcal{C}$ would be correct on $\mathcal{D}$
- $\mathcal{C}$ will be compact enough to be loaded in memory

K-Means for large datasets

Problem

- Learn a global model from a dataset that does not fit in memory and arrives as a stream.

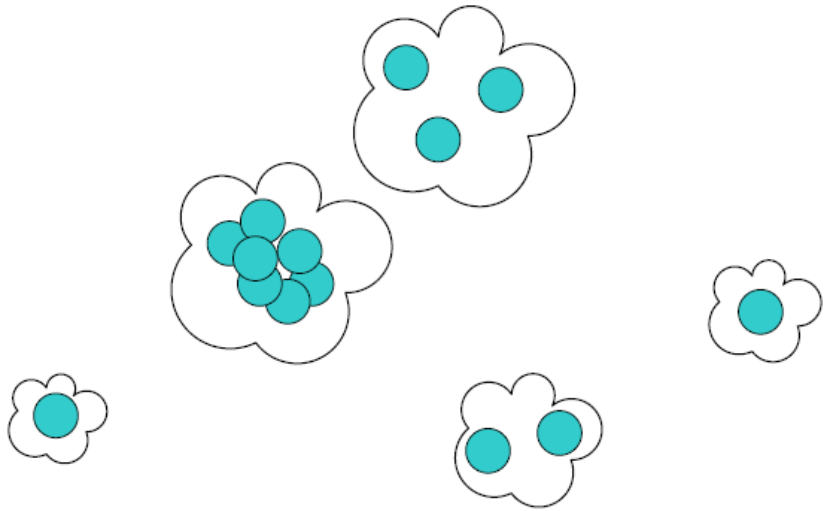
Idea

- Efficiently build a compact version $\mathcal{C}$ of the data $\mathcal{D}$ using sampling (can be handled with streams) such that K-Means results on $\mathcal{C}$ would be correct on $\mathcal{D}$
- $\mathcal{C}$ will be compact enough to be loaded in memory

Solution

- Build $\mathcal{C}$ so that the centroids of $\mathcal{C}$ and $\mathcal{D}$ will be very similar : Use compression based on weighted data representatives

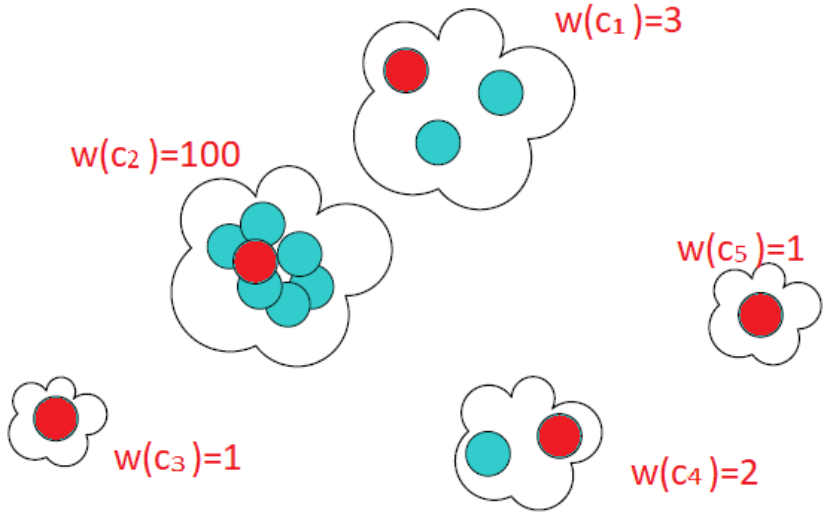
Compressing data for K-Means (1/2)



- **Core idea :** Replace many points by a weighted representative
- New Objective function :

$$L_k(\mu; \mathcal{C}) = \sum_{(w,\mathbf{x})\in\mathcal{C}} w \operatorname*{argmin}_{j\in\{1\cdots k\}} ||\mu_j - \mathbf{x}||_2^2$$

Compressing data for K-Means (2/2)

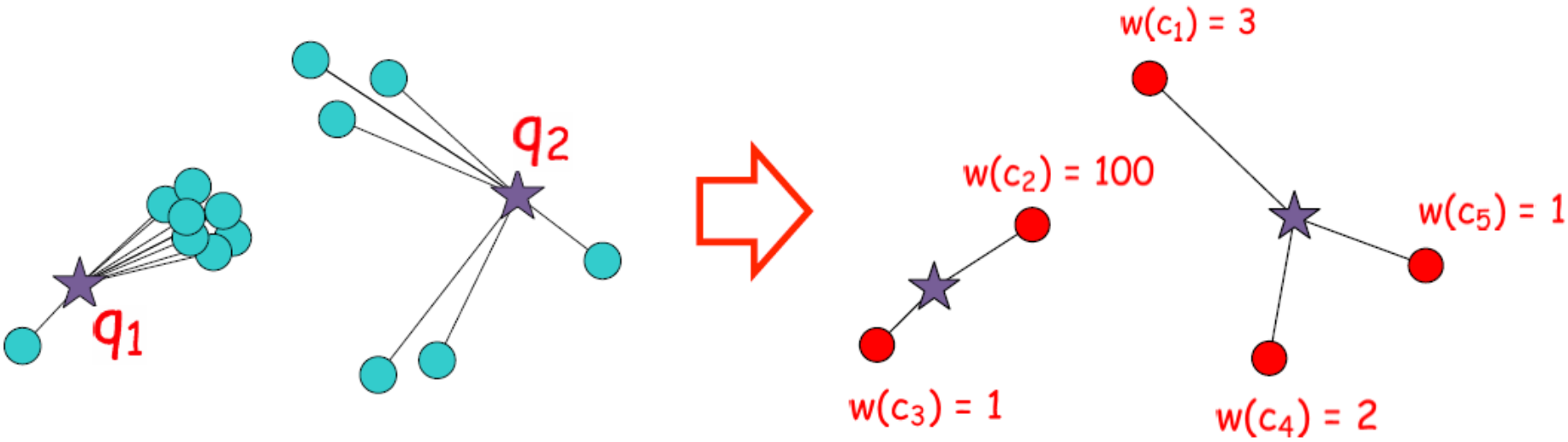


- **Core idea :** Replace many points by a weighted representative
- New Objective function :

$$L_k(\mu; \mathcal{C}) = \sum_{(w,\mathbf{x})\in\mathcal{C}} w \operatorname*{argmin}_{j\in\{1\cdots k\}} ||\mu_j - \mathbf{x}||_2^2$$

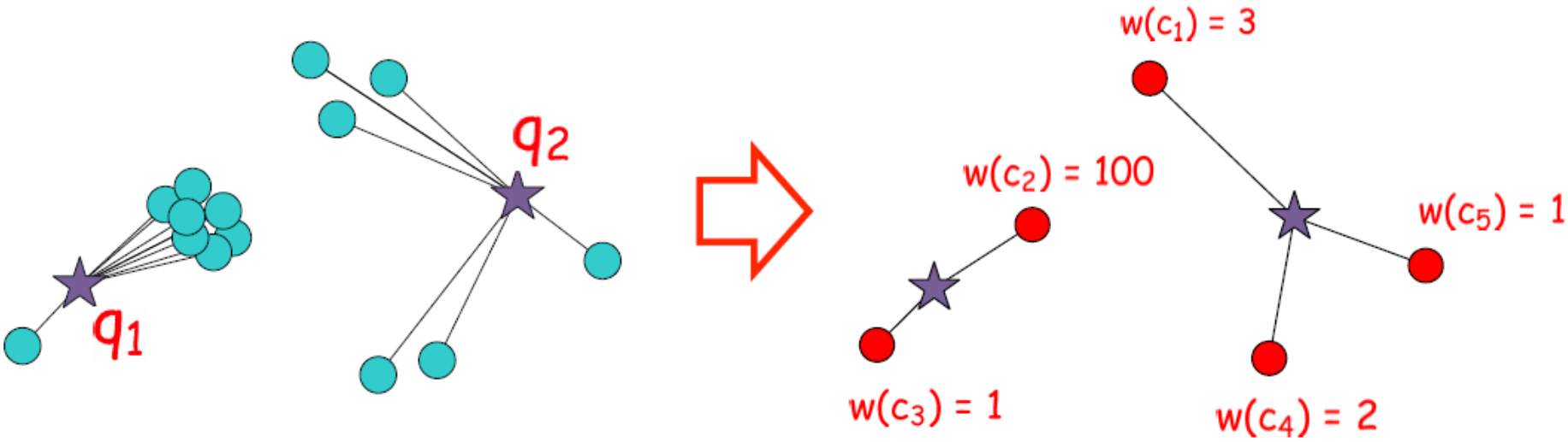
Coresets : Run K-Means on the representatives !

$$L_k(\mu; \mathcal{C}) = \sum_{(w,\mathbf{x})\in\mathcal{C}} w \operatorname*{argmin}_{j\in\{1\cdots k\}} ||\mu_j - \mathbf{x}||_2^2$$



Coresets : Run K-Means on the representatives !

$$L_k(\mu; \mathcal{C}) = \sum_{(w,\mathbf{x})\in\mathcal{C}} w \operatorname*{argmin}_{j\in\{1\cdots k\}} ||\mu_j - \mathbf{x}||_2^2$$



Coreset : Definition

$\mathcal{C}$ is a **(k,ϵ)-coreset** of the dataset $\mathcal{D}$ if :

$$(1-\epsilon)L_k(\mu; \mathcal{D}) \leq L_k(\mu; \mathcal{C}) \leq (1+\epsilon)L_k(\mu; \mathcal{D})$$

Building Coresets

- Suppose that : $\forall(i,j) \quad ||x_i - x_j||_2 \leq 1$
- Random sampling :
  - Pick $n << N$ points $\mathcal{C}$ uniformly at random from $\mathcal{D}$

Building Coresets

- Suppose that : $\forall(i,j) \quad ||x_i - x_j||_2 \leq 1$
- Random sampling :
  - Pick $n << N$ points $\mathcal{C}$ uniformly at random from $\mathcal{D}$

$$\mathbb{E}[L_k(\mu;\mathcal{C})] = L_k(\mu;\mathcal{D})$$

Building Coresets

- Suppose that : $\forall(i,j) \quad ||x_i - x_j||_2 \leq 1$
- Random sampling :
  - Pick $n << N$ points $\mathcal{C}$ uniformly at random from $\mathcal{D}$

$$\mathbb{E}[L_k(\mu;\mathcal{C})] = L_k(\mu;\mathcal{D})$$

  - Using Hoeffding's inequality, we have :

$$Pr(|L_k(\mu;\mathcal{C}) - L_k(\mu;\mathcal{D})| \geq \epsilon) \leq 2\exp(-2\epsilon^2 n)$$

Building Coresets

- Suppose that : $\forall(i,j) \quad ||x_i - x_j||_2 \leq 1$
- Random sampling :
  - Pick $n << N$ points $\mathcal{C}$ uniformly at random from $\mathcal{D}$

$$\mathbb{E}[L_k(\mu;\mathcal{C})] = L_k(\mu;\mathcal{D})$$

  - Using Hoeffding's inequality, we have :

$$Pr(|L_k(\mu;\mathcal{C}) - L_k(\mu;\mathcal{D})| \geq \epsilon) \leq 2\exp(-2\epsilon^2 n)$$

  - For $\mathcal{C}$ to be a (k,ϵ)-coreset with a probability of at least $1-\delta$, it needs at least

$$n = \mathcal{O}\left(\frac{1}{\epsilon^2}\log\frac{1}{\delta}\right) \quad \text{points}$$

Building Coresets : We can do better

Theorem [Har-Peled and Kushal,2005]

One can find efficiently a (k,ϵ)-coreset for K-means of size $\mathcal{O}\left(\frac{k^3}{\epsilon^{\delta+1}}\right)$

Building Coresets : We can do better

Theorem [Har-Peled and Kushal,2005]

One can find efficiently a (k,ϵ)-coreset for K-means of size $\mathcal{O}\left(\frac{k^3}{\epsilon^{\delta+1}}\right)$

Theorem [Feldman et al.,2007]

One can find efficiently a **weak** (k,ϵ)-coreset for K-means of size $\mathcal{O}\left(\text{poly}(k, \frac{1}{\epsilon})\right)$

Coresets on streams : building coresets as data arrive

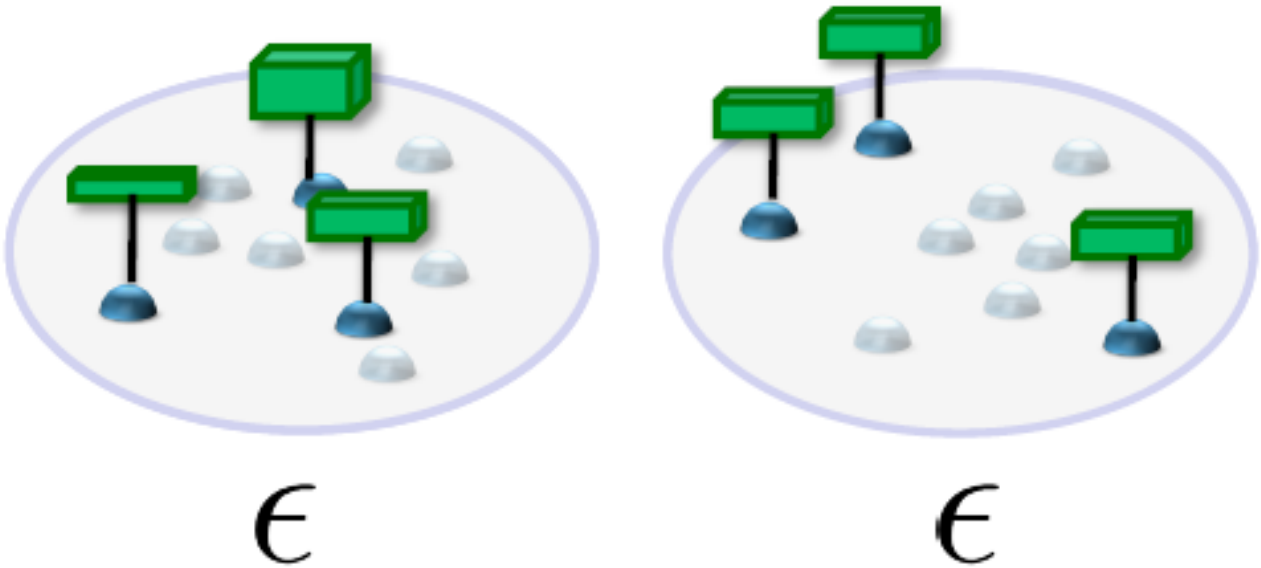
Coresets properties [Har-Peled and Mazumdar,2004]

- **Merge** : The union of two (k,ϵ)-coresets is a (k,ϵ)-coreset.

Coresets on streams : building coresets as data arrive

Coresets properties [Har-Peled and Mazumdar,2004]

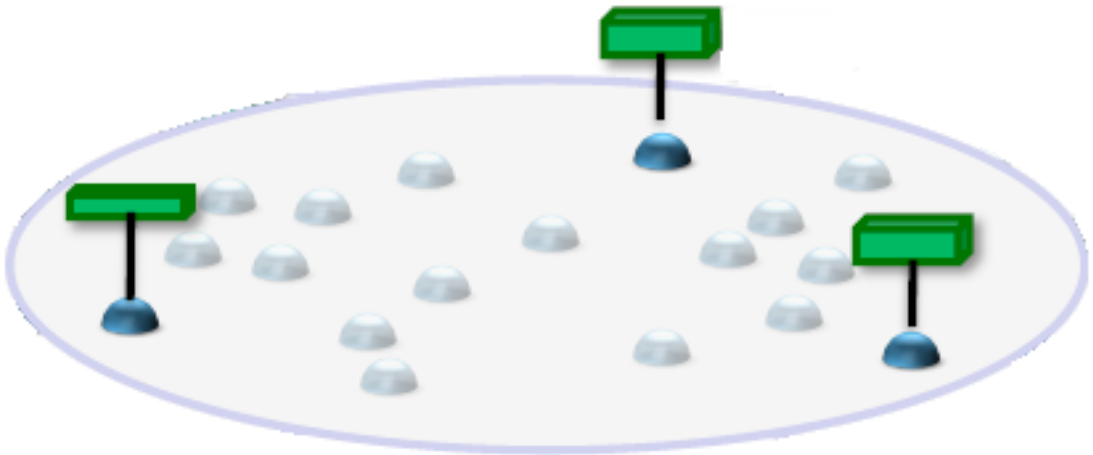
- **Merge** : The union of two (k,ϵ)-coresets is a (k,ϵ)-coreset.
- **Compress** : A (k,δ)-coreset of a (k,ϵ)-coreset is a (k,$\epsilon + \delta + \epsilon\delta$)-coreset.



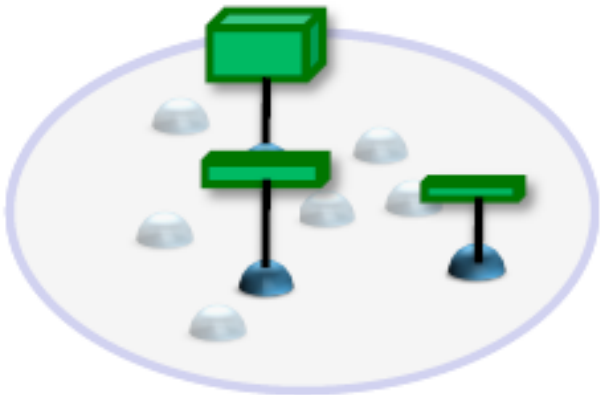
Coresets on streams : building coresets as data arrive

Coresets properties [Har-Peled and Mazumdar,2004]

- **Merge** : The union of two (k,ϵ)-coresets is a (k,ϵ)-coreset.
- **Compress** : A (k,δ)-coreset of a (k,ϵ)-coreset is a (k,$\epsilon + \delta + \epsilon\delta$)-coreset.



$2\epsilon + \epsilon^2$

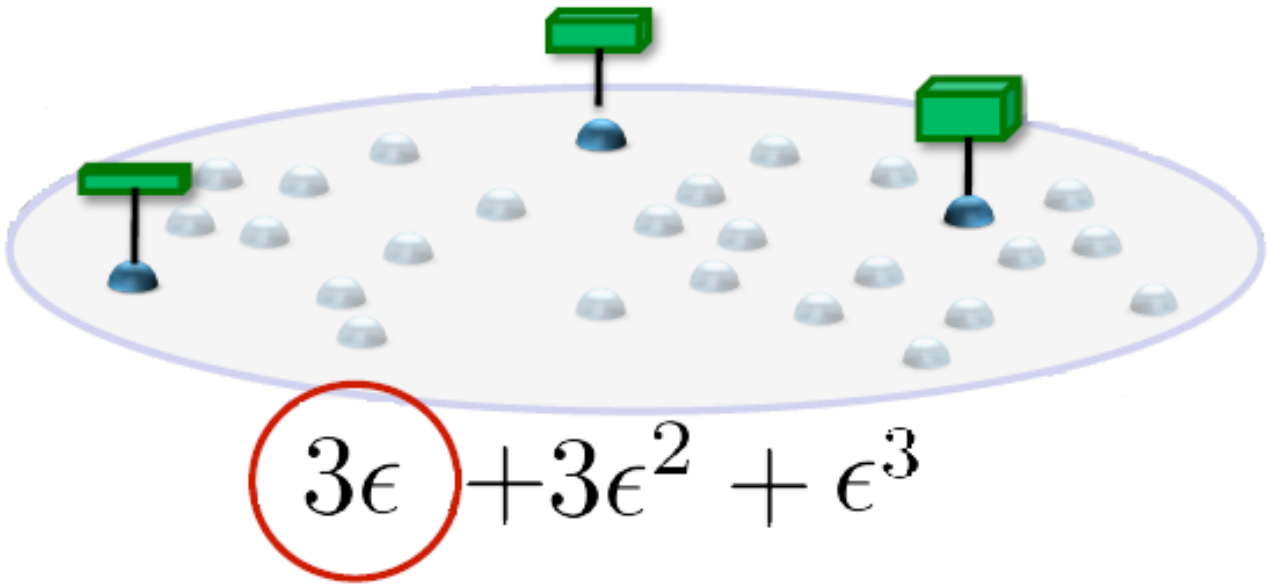


ϵ

Coresets on streams : building coresets as data arrive

Coresets properties [Har-Peled and Mazumdar,2004]

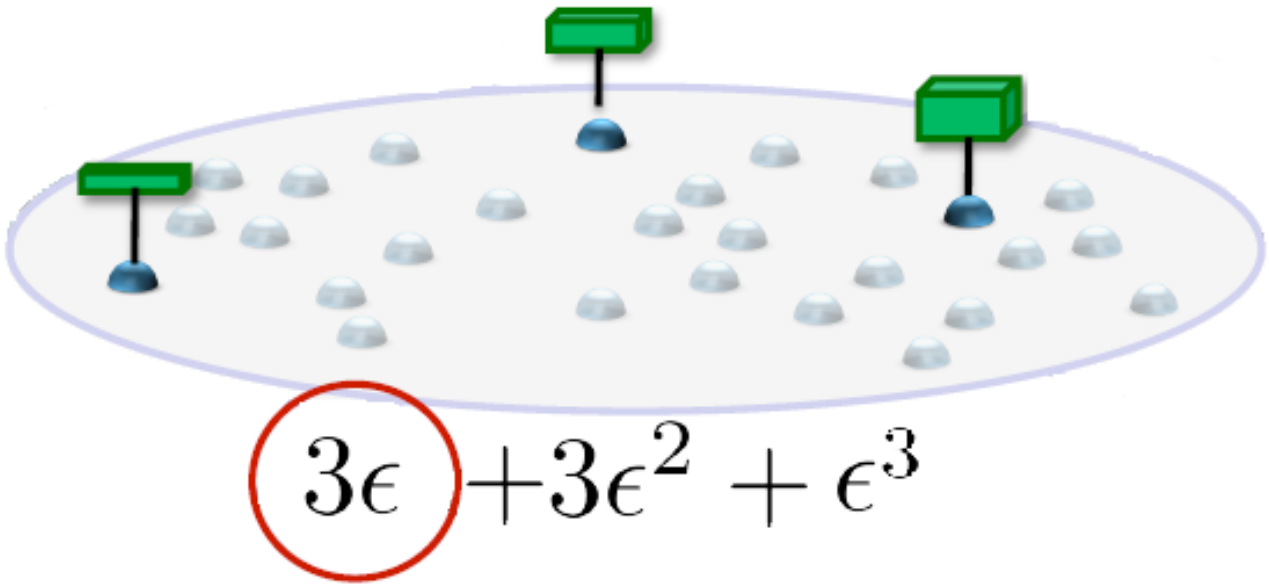
- **Merge** : The union of two (k,ϵ)-coresets is a (k,ϵ)-coreset.
- **Compress** : A (k,δ)-coreset of a (k,ϵ)-coreset is a (k,$\epsilon + \delta + \epsilon\delta$)-coreset.



Coresets on streams : building coresets as data arrive

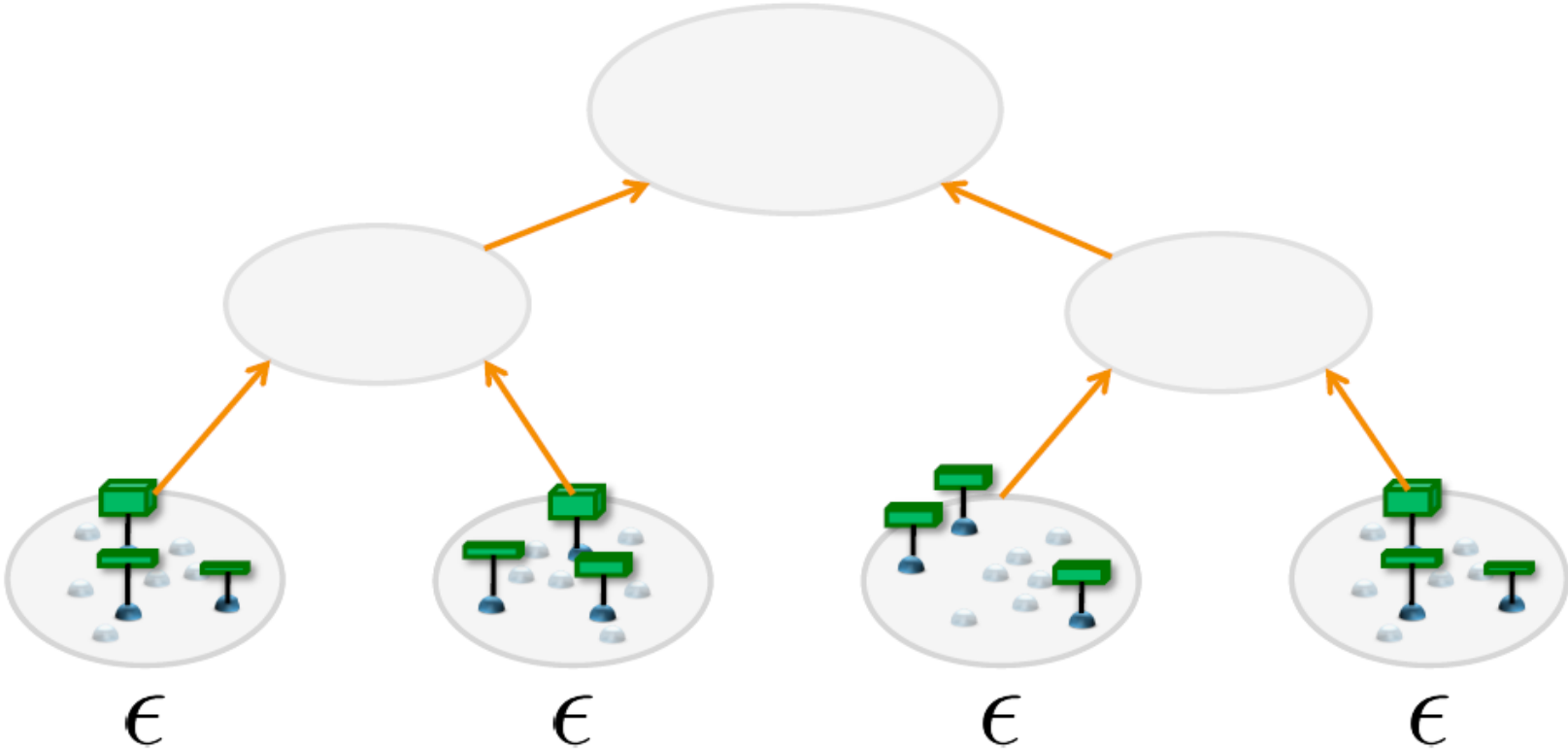
Coresets properties [Har-Peled and Mazumdar,2004]

- **Merge** : The union of two (k,ϵ)-coresets is a (k,ϵ)-coreset.
- **Compress** : A (k,δ)-coreset of a (k,ϵ)-coreset is a (k,$\epsilon + \delta + \epsilon\delta$)-coreset.



The error grows linearly with the number of compressions

Coresets on streams : building coresets as data arrive



The error grows with the depth of the compression tree

K-Means clustering with coresets : Algorithm

Let us consider a data set $\mathcal{D}$, a desired number of clusters k and a precision ϵ.

- Construct the (k,ϵ)-coreset $\mathcal{C}$
  - For example in parallel using MapReduce
  - Or just on the fly as data arrive
- Run K-Means on the coreset $\mathcal{C}$
  - If the coreset is small, we can even do an exhaustive search !
  - In practice : Run K-Means multiple times with many restarts.

K-Means clustering with coresets : Algorithm

Let us consider a data set $\mathcal{D}$, a desired number of clusters k and a precision ϵ.

- Construct the (k,ϵ)-coreset $\mathcal{C}$
  - For example in parallel using MapReduce
  - Or just on the fly as data arrive
- Run K-Means on the coreset $\mathcal{C}$
  - If the coreset is small, we can even do an exhaustive search !
  - In practice : Run K-Means multiple times with many restarts.
- The resulting solution will be $(1+\epsilon)$-optimal for $\mathcal{D}$
- Probably a near-optimal solution !

Remark on Coresets

Coresets can be used with many other learning algorithms :

- K-Means, K-Median
- PageRank
- SVMs
- Matrix low-rank approximation
- ...

Remark on Coresets

Coresets can be used with many other learning algorithms :

- K-Means, K-Median
- PageRank
- SVMs
- Matrix low-rank approximation
- ...

Limits of Coresets

Coresets relies on **random sampling** ! If the datastream is not so random, things can go wrong very quickly !

Outline

Summary so far

- Clustering is a central problem in unsupervised learning and is an even more difficult one with large datasets

Summary so far

- Clustering is a central problem in unsupervised learning and is an even more difficult one with large datasets
- Two main classes of approaches :
  - The Online approach for real time temporal data, eventually with windowing techniques.
  - The compression approach for Offline large data

Summary so far

- Clustering is a central problem in unsupervised learning and is an even more difficult one with large datasets
- Two main classes of approaches :
  - The Online approach for real time temporal data, eventually with windowing techniques.
  - The compression approach for Offline large data
- Discussion on the K-Means algorithm for datastreams
  - Widely used algorithm, efficient with complex data
  - Can scale to large data sets using online learning or coreset constructions

Acknowledgments

These slides are partly based on material by Christopher M. Bishop, Andrew Moore, Danny Feldman, Andreas Krause and Antoine Cornuéjols.

Homeworks for next week

- Find a team mate. You are going to work in **teams of 2 students**.
- Pick a programming language you are familiar with : Python, Java, Matlab, or R.
- Find an implementation of the **regular K-Means algorithm** that can be modified : All functions must be detailed.
  - If you can't find any code that you can modify, make your own version.
  - Do Lab 2 at home for February 28th

Article presentations

- 8 articles
- 36 students
  - 8 teams of 4 to 5 students.
  - It counts for 30% of your final mark.

What you have to do

After picking an article, you will have to prepare a **15 minutes presentation** that will be followed by **5 minutes of questions** from your colleagues and your professors.