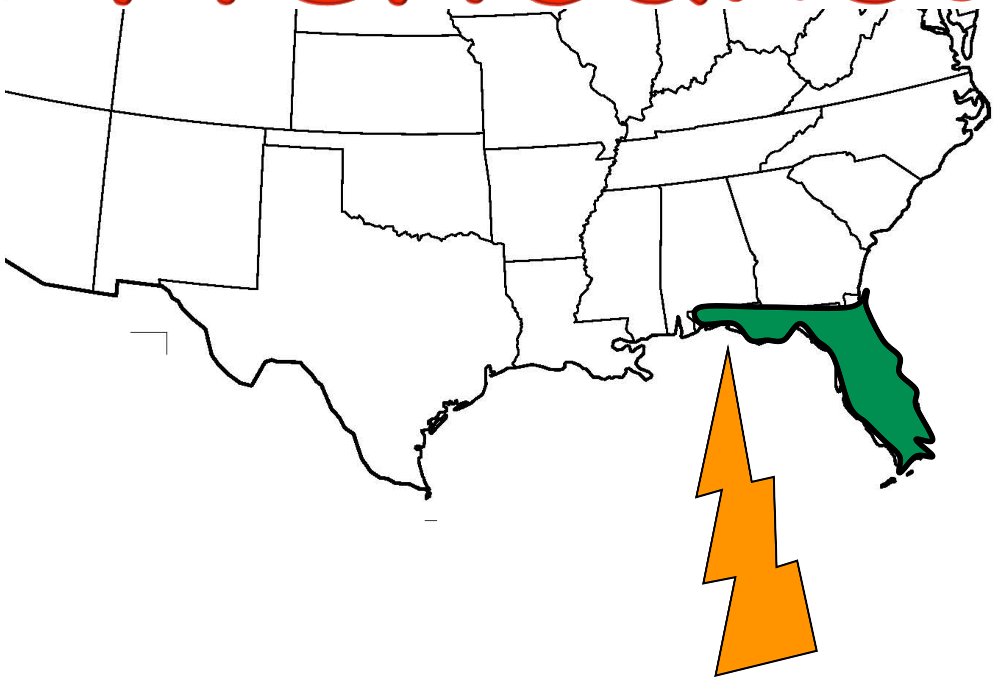
# Аз съм Джим Bezdek

Redfish, Pensacola, 2012

# Здравей и добре дошъл!

# Аз живея в Пенсакола

Amberjack
Pensacola, FL



# I work at the University of Melbourne


*Mudshark*
**Melbourne, Au**


*Wahoo and Ulua*
**Maui, Hawaii**

Sockeye Salmon, Seattle, 1983

Today's Talk: Cluster Analysis in BIG DATA

I. BIG data HCM/FCM/GMD

HCM/FCM/EM~GMD

c-means for BIG data

II. SL + siVAT for BIG data

sVAT visual assessment

clusiVAT algorithm

# Our Big Data Gang

Pal

Palani

Rao

Leckie

Huband

Kumar

Hall


Hathaway

Bezdek

Suthar

| | |
|---|---|
| Complexity reduction for large image processing, *IEEE SMC*, 2002. | Scalable visual assessment of cluster tendency for large data sets, *Pattern Recognition*, 2006. |
| Extending fuzzy and probabilistic clustering to VL data sets, *Comp. Stat. And Analysis*, 2006. | Fuzzy *c*-Means Algorithms for Very Large Data, *IEEE TFS*, 2012. |
| Approximate clustering in very large relational data, *IJIS*, 2006. | A hybrid approach to clustering in big data, *IEEE Trans. Cybernetics*, 2016. |

## 2 Kinds of *Basic Numerical Data* for Pattern Recognition

**Objects** $O = \{o_1, \ldots, o_n\}$ : $o_i$ = i-th *physical* object

**Object Data** $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^p$ : $x_i$ = *feature vector* for $o_i$

$x_{ji}$ = j-th (*measured*) feature of $x_i$ : $1 \le j \le p$

| |
|---|
| **Sizes** : n = # samples; p = # dimensions |

**Relational Data** $R = [r_{ij}]$ = *relationship* $(o_i, o_j)$ ) or $(x_i, x_j)$

$d_{ij}$

$s_{ij}$ = pairwise *similarity* $(o_i, o_j)$ or $(x_i, x_j)$

$d_{ij}$ = pairwise *dissimilarity* $(o_i, o_j)$ or $(x_i, x_j)$

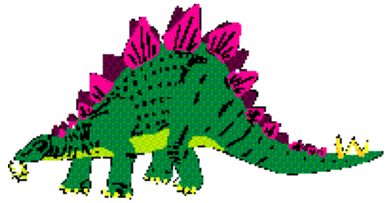
**Typically (R = D)**

$$\left.\begin{array}{ll} d_{ii} = 0 & : 1 \le i \le n \\ d_{ij} > 0 & : 1 \le i, j \le n \end{array}\right\} \text{(Positive-definite)}$$

$$d_{ij} = d_{ji} \quad : 1 \le i \ne j \le n \quad \text{(Symmetric)}$$

We often convert X→D with *distance* $d_{ij} = \left\| x_i - x_j \right\|$

# How Big *is* static BIG Data ?

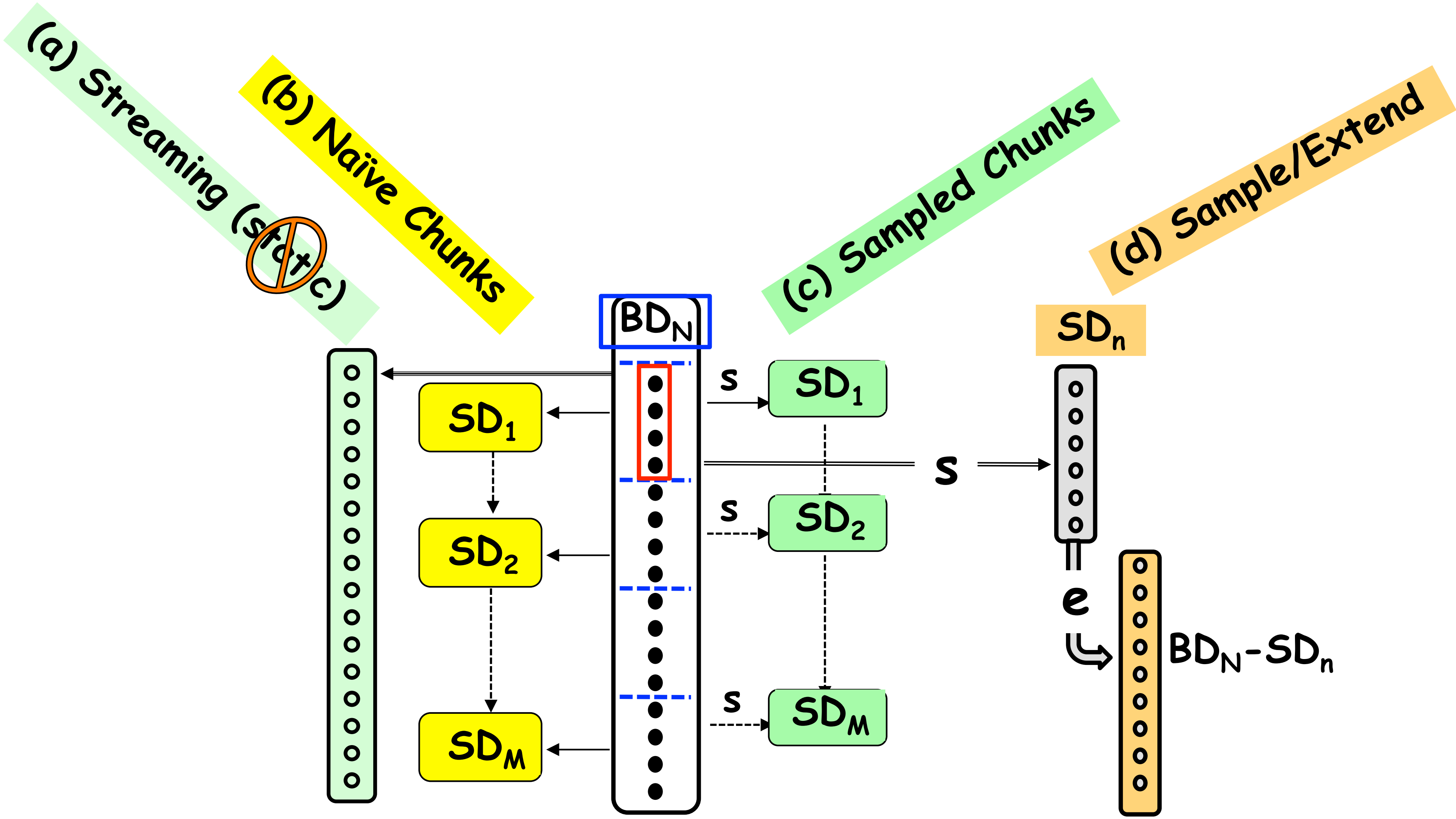


| Bytes | *Big Data* (BD) in size (n, p) |
|---|---|
| $10^3$ | Kilo – small |
| $10^6$ | Mega - medium |
| $10^9$ | Giga - large |
| $10^{12}$ | Terra – getting' up there |
| $10^{15}$ | Peta - MONSTER |
| $10^{>18}$ | Exa –Yikes ! (BIG DATA) |

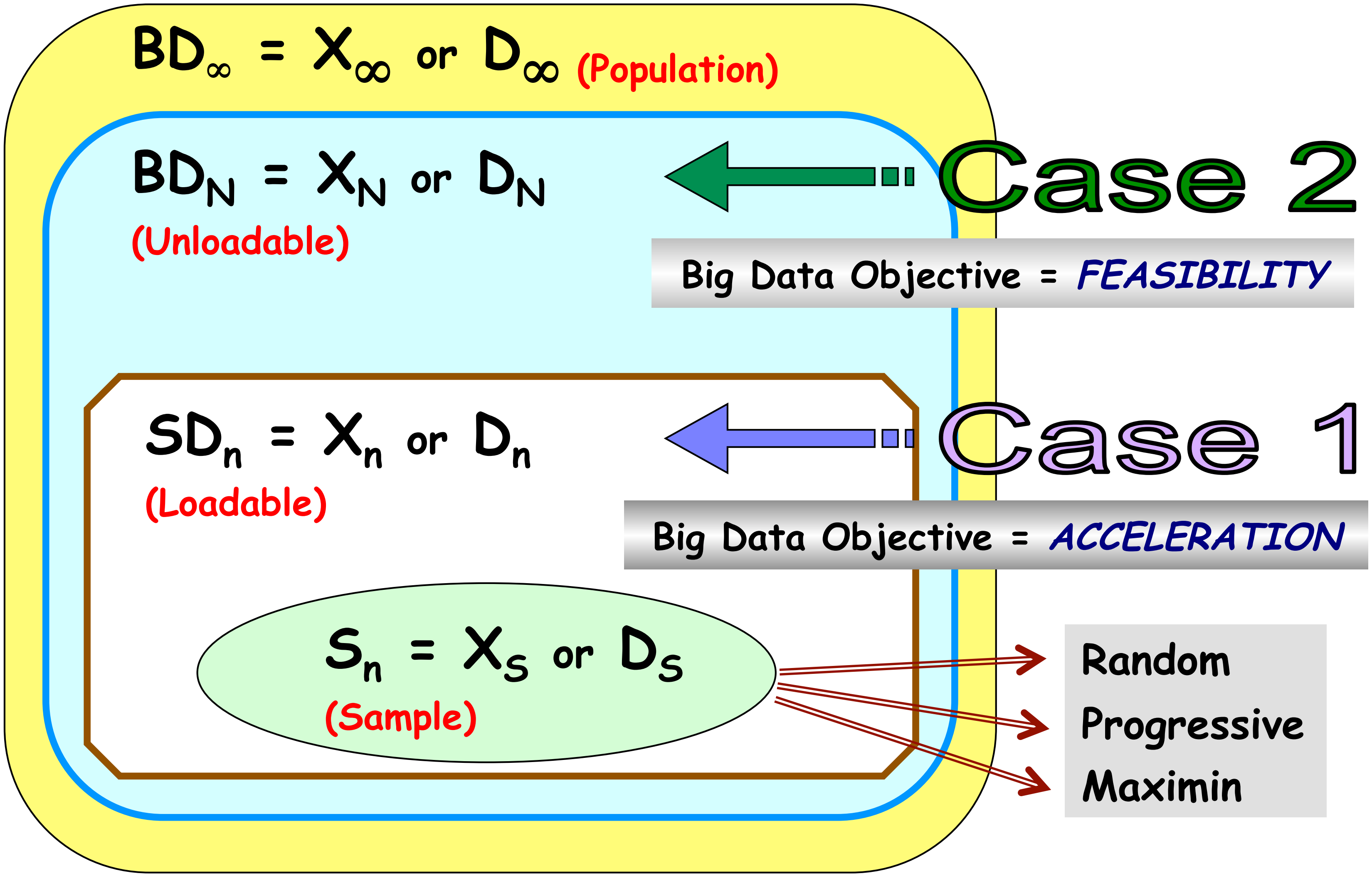
We can't cluster or image data this big (in a single computer)… so …

Most BIG data methods build "cluster-friendly" (loadable) subsets by *sampling* or *chunking*

# 4 ways to make static Big ($BD_N$) Data Small(er)



# Four Data Levels



Case 1

If $X_n$ or $D_n$ is *Loadable* we can *Quantitatively* *Compare* Lit-Clusters ↔ Approx. Clusters

Case 2

If $X_N$ or $D_N$ is *Unloadable* *Comparison is impossible*

so …

case 2 validity rests with "good" case 1 examples

Case 2 BD methods often provide acceleration *and* feasibility

# Case 2

**If $X_N$ or $D_N$ is *Unloadable* …**

**there are 3 basic approaches to scaling up**

1. Cluster a sample, non-iterative extension
2. Incremental/Distributed data clustering
3. Kernel-based methods, e.g., kFCM

Many of these methods can be used with other pattern recognition algorithms. For clustering, we let …

A*= [U*, V*] = exact (literal) partition and prototypes
A = [U, V] = (any) approximation to A* by (1) or (2)

row i ➠ M'ship of all $o_k$'s in cluster i

$o_1$ $o_k$ $o_n$ Objects

$$U = \begin{bmatrix} u_{11} & \cdots & u_{1k} & \cdots & u_{1n} \\ \vdots & & \vdots & & \vdots \\ u_{i1} & \cdots & u_{ik} & \cdots & u_{in} \\ \vdots & & \vdots & & \vdots \\ u_{c1} & \cdots & u_{ck} & \cdots & u_{cn} \end{bmatrix}$$

clusters

col k ➠ M'ship of $o_k$ in each cluster

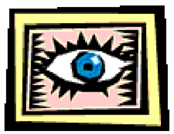
## Partition Matrices

Membership Functions

$u_i : O \rightarrow [0,1]$

$u_i(o_k) = u_{ik}$ = M'ship of $o_k$ in cluster i

| | Crisp | | Fuzzy/Prob | | Possibilistic |
|---|---|---|---|---|---|
| Row sums | $\sum_{k} u_{ik} > 0$ | | same | | same |
| Col sums | $\sum_{i} u_{ik} = 1$ | | same | | $\sum_{i=1}^{c} u_{ik} \le c$ |
| M'ships | $u_{ik} \in \{0,1\}$ | | $u_{ik} \in [0,1]$ | | same |
| Set Name | $M_{hcn}$ | $\subset$ | $M_{fcn}$ | $\subset$ | $M_{pcn}$ |
| Example | $\begin{matrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{matrix}$ | | $\begin{matrix} 1 & .07 & 0 & .44 \\ 0 & .91 & 0 & .06 \\ 0 & .02 & 1 & .50 \end{matrix}$ | | $\begin{matrix} 1 & .07 & 1 & .44 \\ 0 & .91 & 0 & .52 \\ 0 & .02 & 1 & .38 \end{matrix}$ |

Take a 2nd 

# Batch Hard and Fuzzy c-Means Models

**Objective function**

$$J_m(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c} u_{ik}^m \left\| x_k - v_i \right\|_A^2$$

$$\left\| x_k - v_i \right\|_A^2 = (x_k - v_i)^T A (x_k - v_i)$$

**Inputs**

Object data $X = \{x_1, \ldots, x_n\} \subset \Re^p$

**Unknowns**

(Fuzzy) Partition $U \in M_{fcn}$

Prototypes $V = \{v_1, \ldots, v_c\} \in \Re^{cp}$

**Optimization Problem, m≥1**

$$\underset{\substack{U \in M_{fcn} \\ V \in \Re^{cp}}}{\text{minimize}} \left\{ J_m(U,V) = \sum\sum u_{ik}^m \left\| x_k - v_i \right\|_A^2 \right\}$$

# FONCs for extrema of the HCM/FCM Functionals

**FCM** | **HCM**

Prototypes
V=F(U,X)

$$v_i = \frac{\sum_{k=1}^{n} (u_{ik})^m x_k}{\sum_{j=1}^{n} (u_{ij})^m}$$

limit=
$m \rightarrow 1^+$

$$v_i = \frac{\sum_{k=1}^{n} u_{ik} x_k}{\sum_{k=1}^{n} u_{ik}}$$

Partition
U=G(V,X)

$$u_{ik} = \left[ \sum_{j=1}^{c} \left( d_{ikA} \big/ d_{jkA} \right)^{\frac{2}{m-1}} \right]^{-1}$$

limit=
$m \rightarrow 1^+$

$$u_{ik} = \begin{cases} 1 & d_{ikA} \le d_{jk_A},\ j \ne i \\ 0 & \text{otherwise} \end{cases}$$

$$d_{ikA} = \left\| x_k - v_i \right\|_A = \sqrt{(x_k - v_i)^T A (x_k - v_i)}$$

$A_{p \times p}$ positive definite

# Weighted FCM/HCM = wFCM/wHCM

**wFCM**

$$\min_{(U,V)}\left\{J_{mw}(U,V:X)=\sum_{k=1}^{n}\sum_{i=1}^{c}w_k(u_{ik})^m\left\|x_k-v_i\right\|_A^2\right\}$$

**Inputs**

$X \subset \mathbb{R}^p$ + n fixed weights $\{w_k\} \subset (0,\infty)$

**Unknowns**

$U \in M_{fcn}$ + $V=\{v_1,\dots,v_c\} \subset \mathbb{R}^{cp}$

Partition
U=G(V,X)

$$u_{ik}=\left[\sum_{j=1}^{c}\left(d_{ikA}/d_{jkA}\right)^{\frac{2}{m-1}}\right]^{-1}$$

**FONCs**

Prototypes
V=F(U,X)

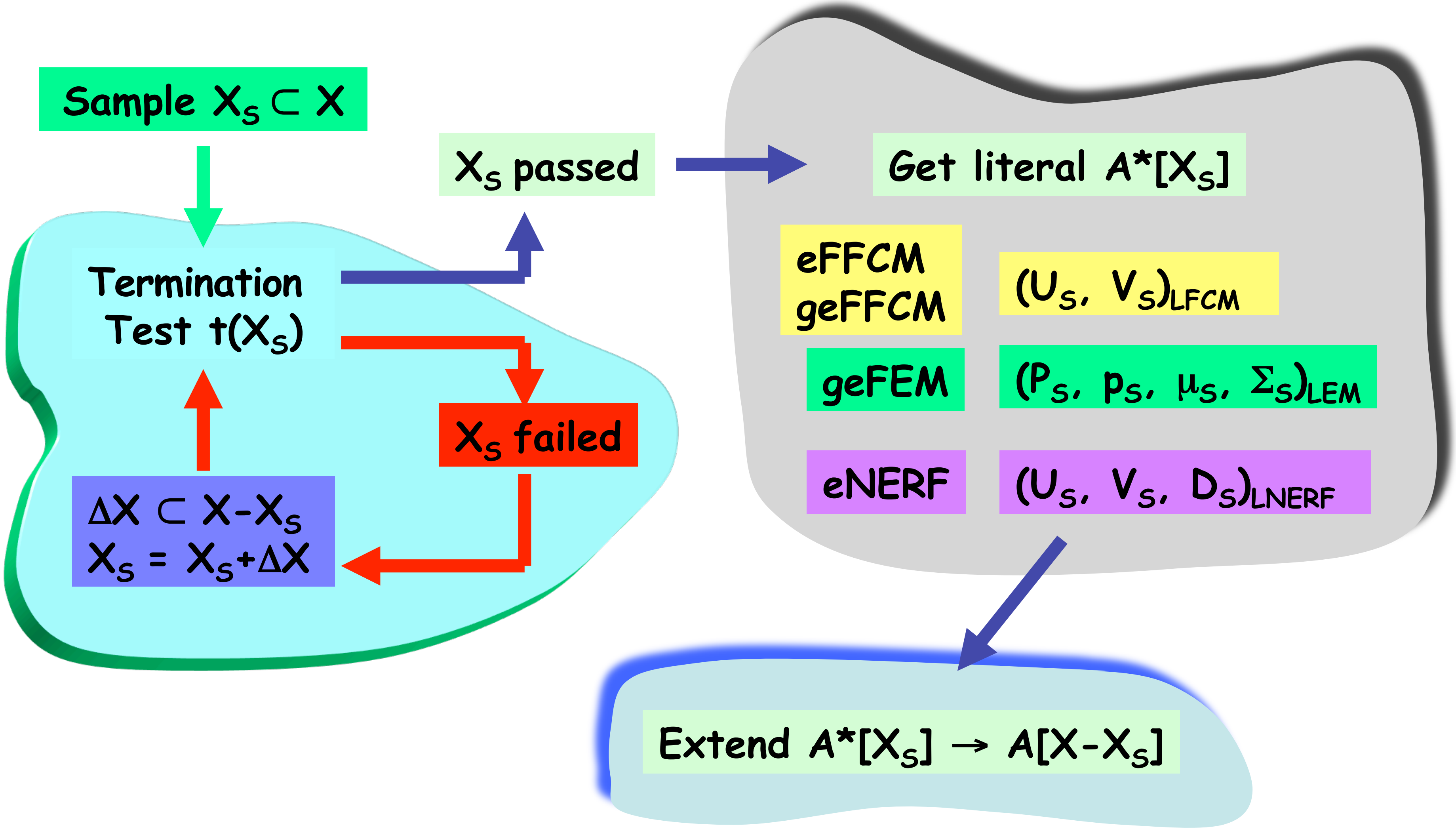
$$v_i=\frac{\sum_{k=1}^{n}w_i(u_{ik})^m x_k}{\sum_{j=1}^{n}w_i(u_{ij})^m}$$

ONLY Change
(Bezdek, 1981)

**BASIC HCM/FCM AO Algorithms**

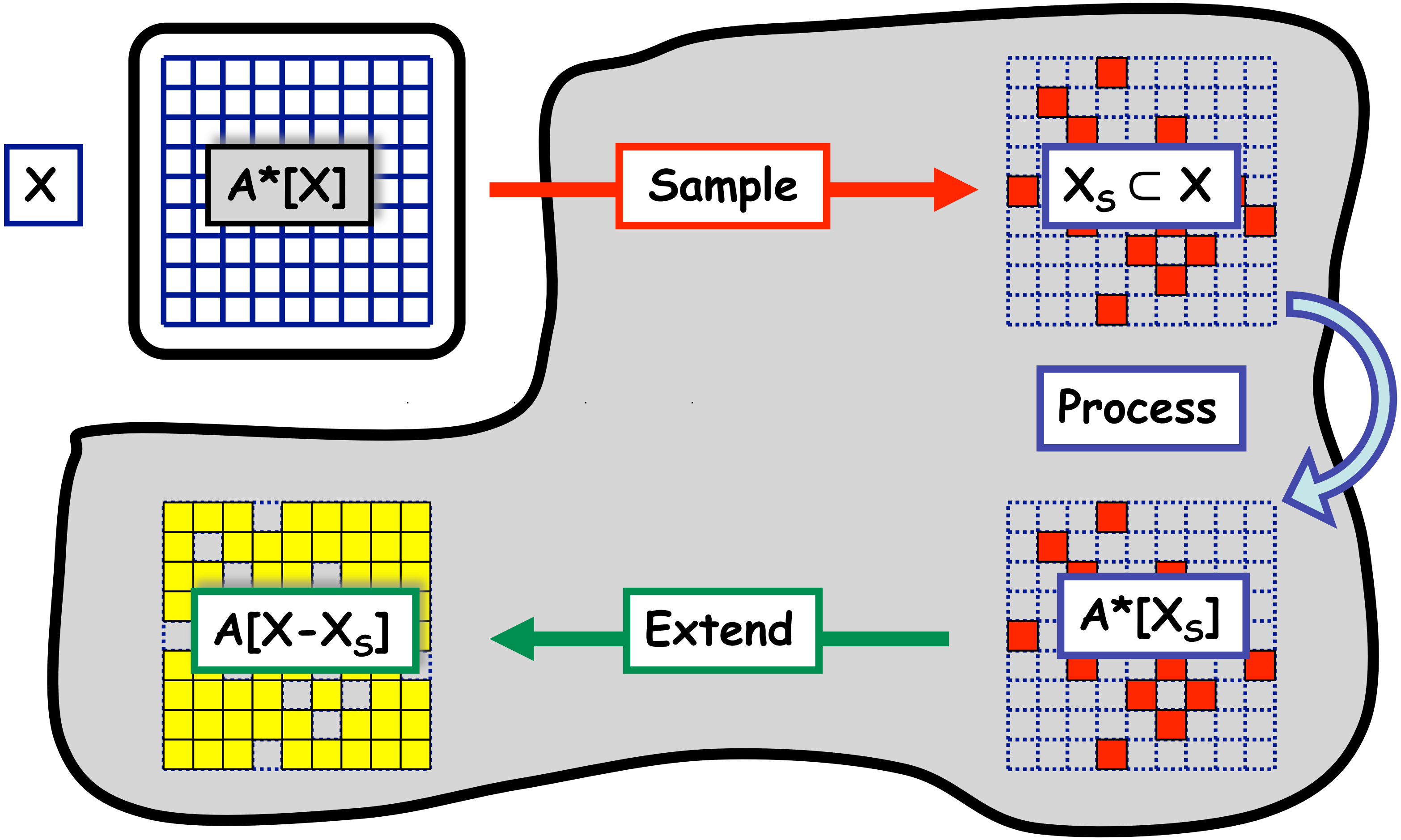
| | |
|---|---|
| Input | *Unlabeled Object data*: $X \subset \mathbb{R}^p$ |
| User Picks | $c,\ m,\ \varepsilon,\ T,\ \left\|*\right\|_A,\ \left\|*\right\|_{err}$ |
| Initialize | $V_0 = (v_{10},\dots,v_{c0}) \in \Re^{cp}$ |
| | $U_0 = G(V_0,X)$ |
| | $V_1 = F(U_0,X)$ --------> % For loop startup |
| | $t=0$ |
| AO Loop | WHILE $[t<T$ and $\left\|V_{t+1} - V_t\right\|_{err} > \varepsilon]$ |
| | $U_{t+1} = G(V_{t+1},X)$ % Next partition |
| | $V_{t+2} = F(U_{t+1},X)$ % Next prototypes |
| | WEND |
| Outputs | $(U^*, V^*) \in M_{fcn} \times \Re^{cp}$ |

# What is *Progressive Sampling* ?



## What is *Extensibility* ?

Algorithm $A : X \subset \Re^p \mapsto A[X] \subset \Re^q$




Literal $A^*[X]$ $\cong$ Approx. $A[X]=A^*[X_S] \| A[X-X_S]$

Sample $X_S \subset X$

Process $FCM[X_S]$

(Non-Iterative) Generalized extension of *Fuzzy c-Means* [FCM ➔ eFFCM/geFFCM]

Note: also works for HCM

Extend $FCM[X_S] \rightarrow FCM[X-X_S]$

with prototypes $V_S$ and $x_k \in X-X_S$

Remark: The extension step usually takes ~ 1% of overall CPU time

$(X - X_S) \ni x_k$

$\{\cdots v_{i,S} \cdots v_{j,S} \cdots\} = V_S$

$$u_{ik} = \left[ \sum_{j=1}^{c} \left( \left\| x_k - v_{i,S} \right\|_A \Big/ \left\| x_k - v_{j,S} \right\|_A \right)^{\frac{2}{m-1}} \right]^{-1} = \varphi(V_S, X - X_S)$$

FONC for U to min $J_m$

Works as a *classifier on X-$X_S$*, but trained w° labels !

Sample $X_S \subset X$

Process $EM[X_S]$

Extend $EM[X_S] \rightarrow EM[X-X_S]$

(Non-Iterative) Generalized Extension of EM (*Gaussian Mixture Decomp.*) [EM → geFEM]

with priors $\{p_{iS}\}$, means $\{\mu_{iS}\}$, covariances $\{\Sigma_{iS}\}$ and $x_k \in X-X_S$

$(X - X_S) \ni x_k$ $\{\mu_{iS}\}$ $\{\Sigma_{iS}\}$

$$g_{ik} = \exp\left(-0.5 * \left\|x_k - \mu_{iS}\right\|^2_{\Sigma_{iS}^{-1}}\right) \Big/ \sqrt{(2\pi)^s \left|\Sigma_{iS}\right|}$$

FONC for U=P to min ln(L)

$$p_{ik} = p_{iS}\, g_{ik} \Big/ \sum_{j=1}^{c} p_{jS}\, g_{jk}$$

$\{p_{iS}\}$

Works as a *classifier on X-XS*, trained w° labels !

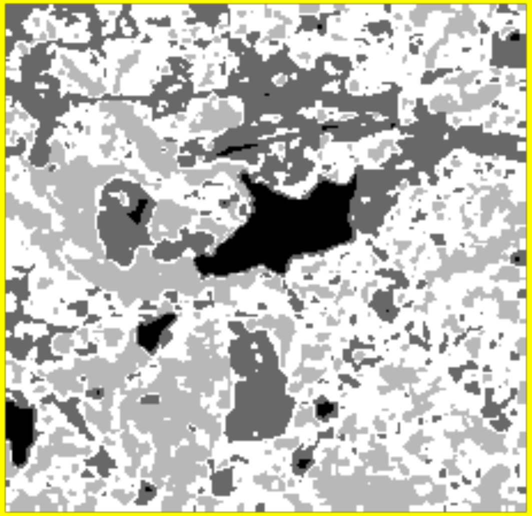
Example: PS + E with FCM on an

Indian Satellite (Very Small Landsat Image)

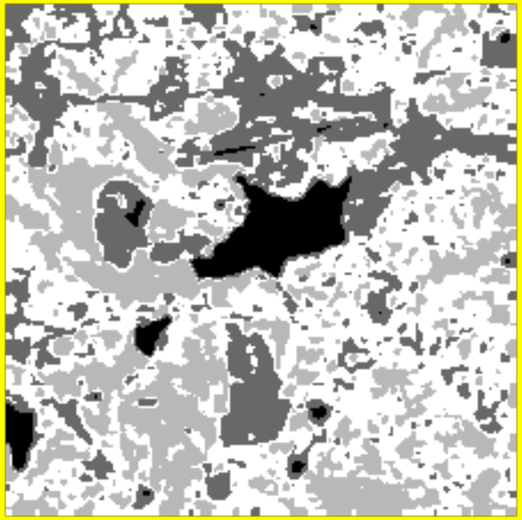
Typical Output Images

COMPARE 9% vs 100% !

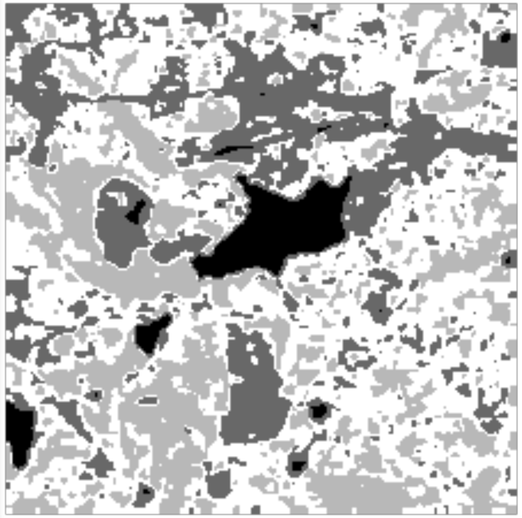
Input image
256x256x256



LFCM 100% of data
mrFCM 100% of data

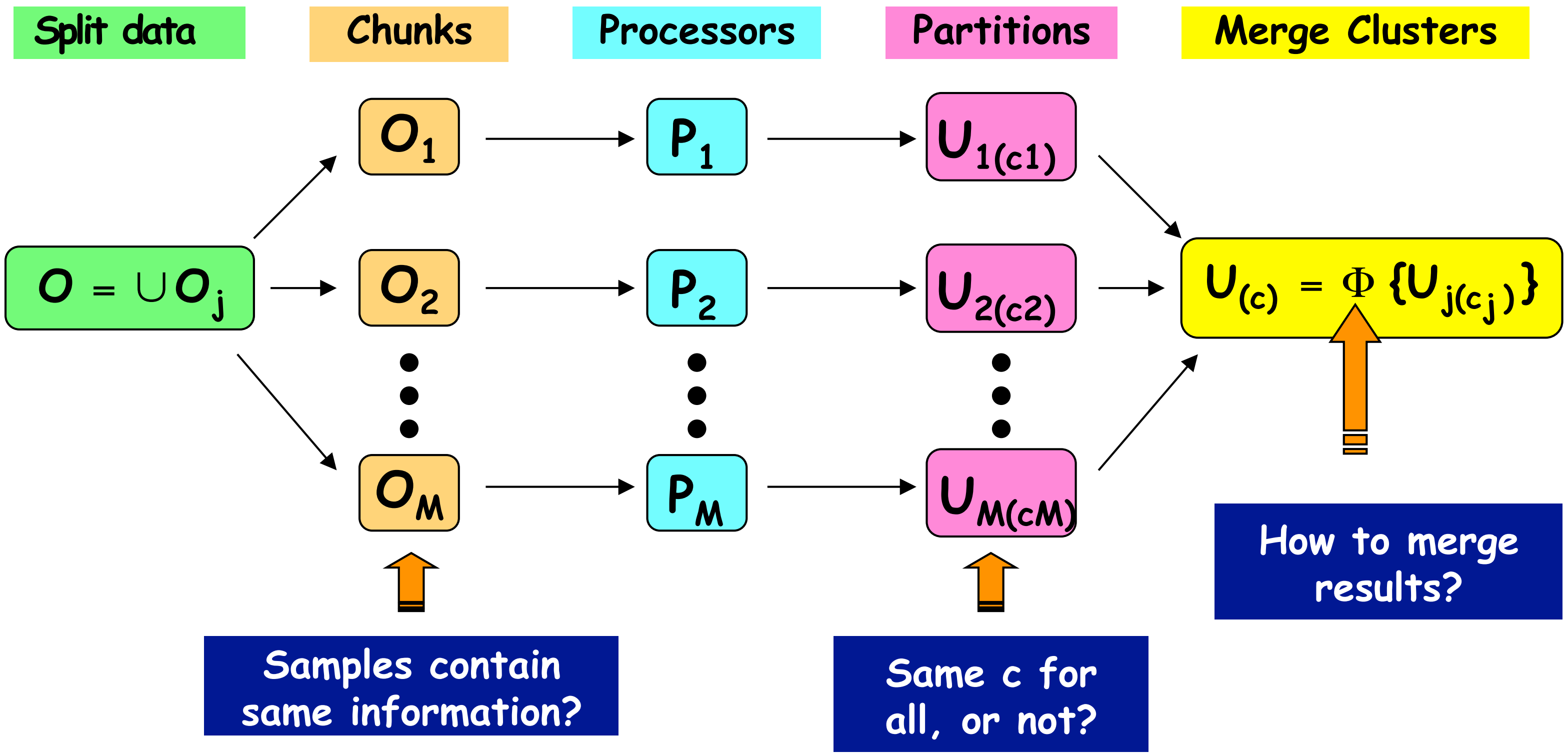


eFFCM
9% of data, div only



eFFCM
29% of data, div & $\chi 2$

# Incremental/Distributed Clustering in *BIG* Data




**3 Problems for the Distributed Clustering Approach**

# Incremental/Distributed c-Means Clustering in VL Data

**brFCM = "bit reduct." FCM**

Eschrich/Ke/Hall/Goldgof (*2003, brFCM*). Fast accurate fuzzy clustering through data reduction. *IEEE TFS*, 11(2), 262–270.

Compression *for image data*, uses wFCM algorithm, (loadable) implementation

**spFCM = "single pass" FCM**

Hore, Hall, Goldgof (*2007, spFCM*). Single pass fuzzy c- means, Proc. FUZZ-IEEE 2007, 1-7.

Uses wFCM algorithm, partially distributed VL implementation

**oFCM = "on line" FCM**

Hore/Hall/Goldgof/Gu/Maudsley/Darkazanli (*2009, oFCM*). A scalable framework for segmenting MRIs, *J Signal Proc. Syst*. 54(1-3), 183–203.

Uses wFCM algorithm, fully distributed VL implementation

**All 3 are generalizations of HCM (k-means) when m = 1**

# spFCM="single pass" and oFCM = "on line" FCM

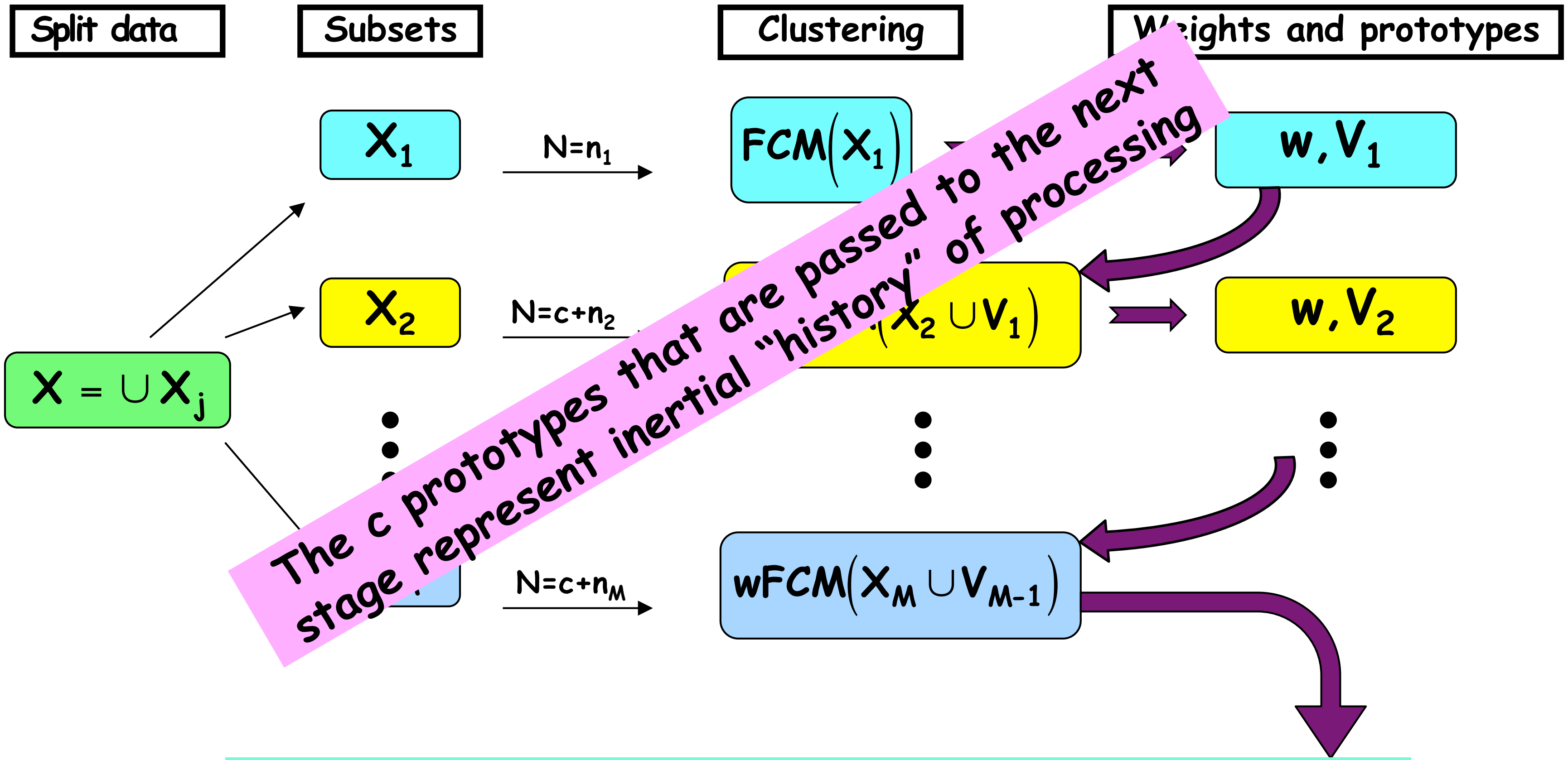
**Split data** $X = \cup X_j \quad : n = \sum n_j$

**First pass** $FCM(X_1) = (U, V)$

**Rowsums of U *after* pass j ≥1** $\omega_i = \sum_{k=1}^{n_j} u_{ik} ; 1 \le i \le c$

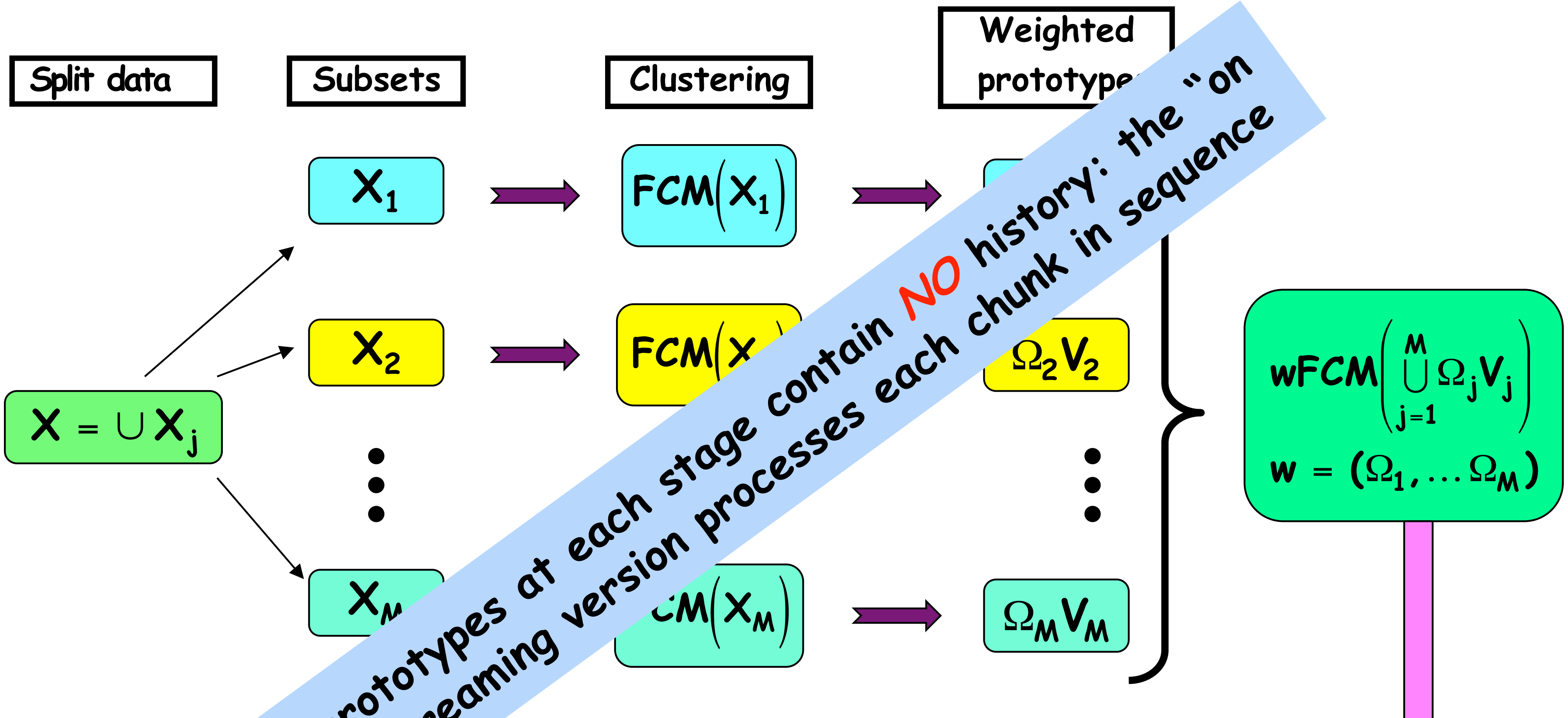
**weights for wFCM *before* pass j>1** $w = ([1], \Omega) = (\underbrace{1, 1, \ldots, 1}_{n_j \text{ times}}, \omega_1, \ldots, \omega_c)$

# Architecture of spFCM : c is chosen and fixed by user




A crisp/fuzzy partition of $X_{VL}$ is built using the HCM/FCM FONCs to compute U with the final V's

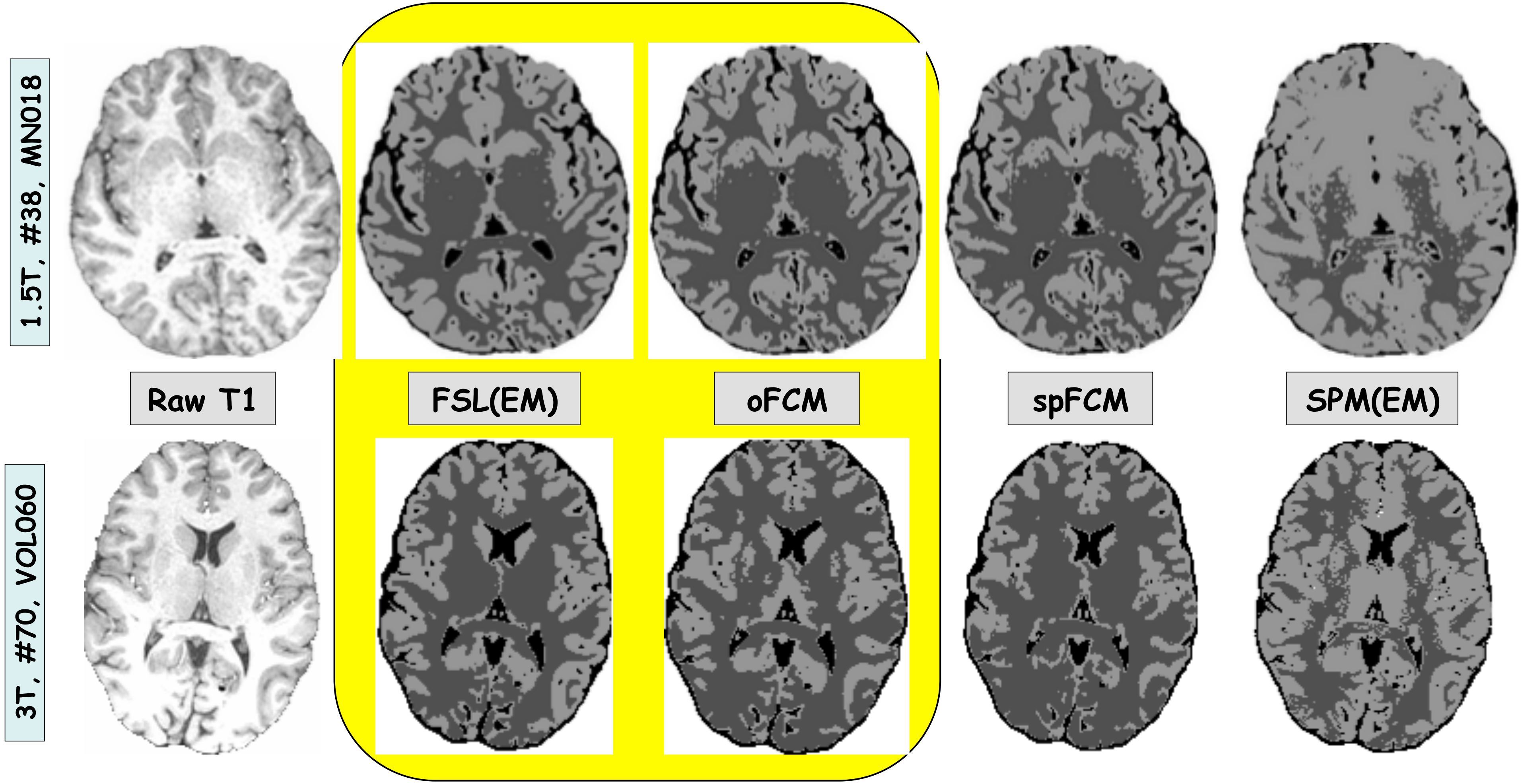
# Architecture of oFCM : c = "max" is same for all blocks




The c prototypes at each stage contain **NO** history: the "on line" or streaming version processes each chunk in sequence

fuzzy partition of $X_{VL}$ is built using
the CM/FCM FONC for U with the final V's

# *Visual comparison* of segmentation with spfcm/ofcm to EM




**FSL ~ oFCM best segment raw images : SPM worst**

# Comparing rseFCM, spFCM & oFCM to LFCM

rseFCM = random sampling + LFCM($X_{ns}$) + extension to $X_N$-$X_{ns}$

LFCM = literal FCM (loadable data) + extension to $X_N$-$X_{ns}$

spFCM = single pass FCM (Hall's model) + extension to $X_N$-$X_{ns}$

oFCM = online FCM (Hall's model) + extension to $X_N$-$X_{ns}$

**3 Evaluation Criteria**

| | |
|---|---|
| *Run time* (time LFCM/time BD Approximation) | 9 data sets |
| *Adjusted Rand Index* ($-\varepsilon \le$ ARI $\le 1$):<br>H(U) = maxcol. hardened U<br>$ARI_1(H(U) \mid U_{GT})$ ⟺ Labeled data<br>$ARI_2(H(U), H(U_{LFCM}))$ ⟺ Unlabeled data | |
| *(Soft)* $ARI_s$ ($-\varepsilon \le ARI_s < 1$) matches approximate/literal fuzzy U's $ARI_s(U, U_{LFCM})$ | 6 unlabeled MRI image data sets |

# MNIST Data: $n$=70,000, $p$=784, $c$=10

**Each Image:** 28 x 28

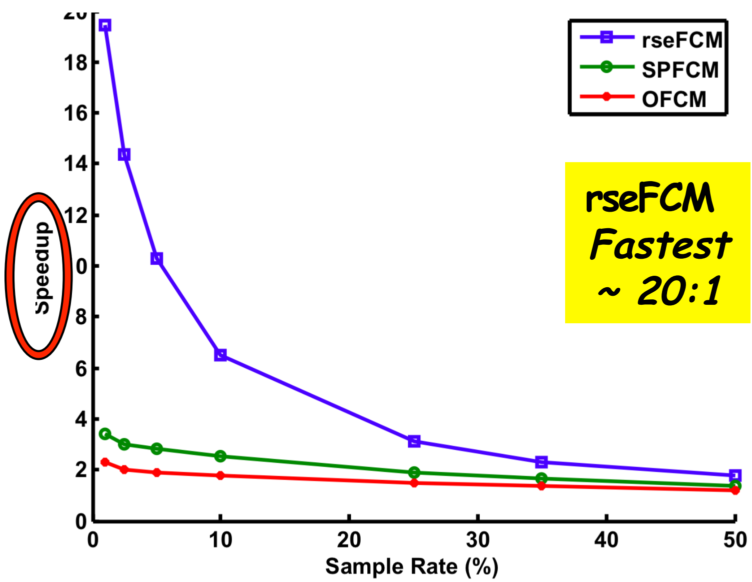
**Each Pixel $p_{ij}$:** 0 to 255

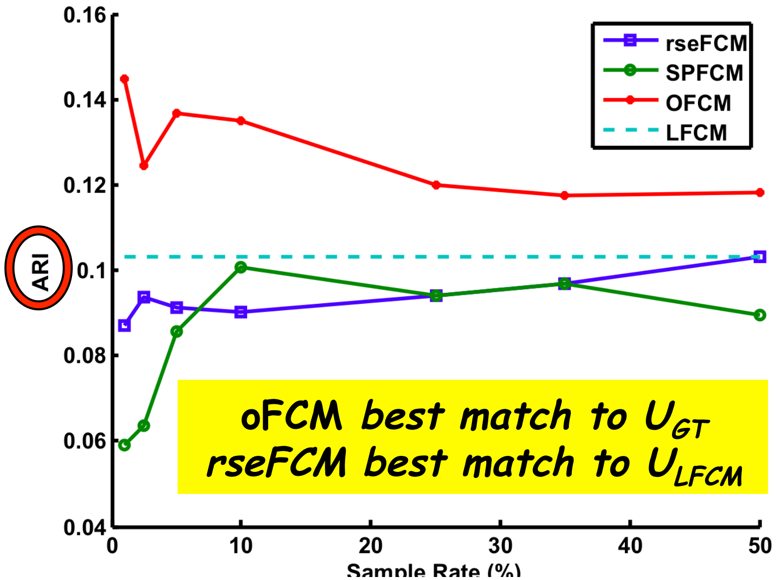
**Each Pixel Normalized:** $(p_{ij}/255)$ so 0 to 1

**Each Vector:** 784 values
$\underline{p} = (p_{11}, \ldots, p_{ij}, \ldots, p_{28,28})$

Presumably $c = 10$, but …
this data does *NOT* cluster well

# MNIST Data n=70,000, p=784, c=10 (Typical Results)

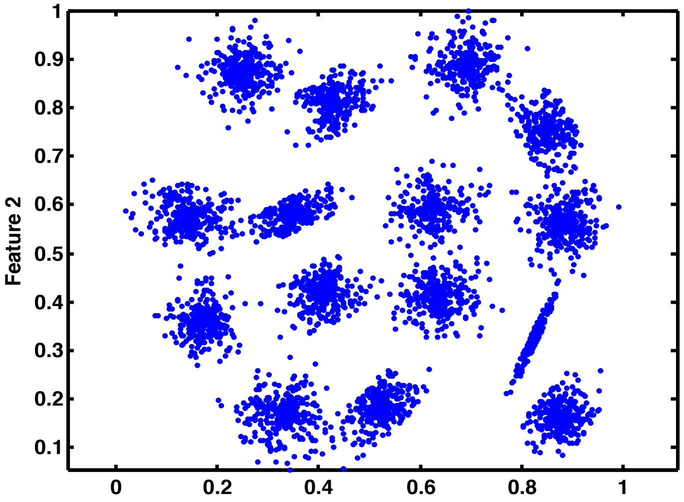



LFCM line (- - - -) is ARI match of $H(U_{LFCM})$ to $U_{GT}$

Other graphs show ARI matching of $H(U_{VL})$ to $U_{GT}$

**Approximation quality of VL-FCMs: compare other graphs to LFCM**

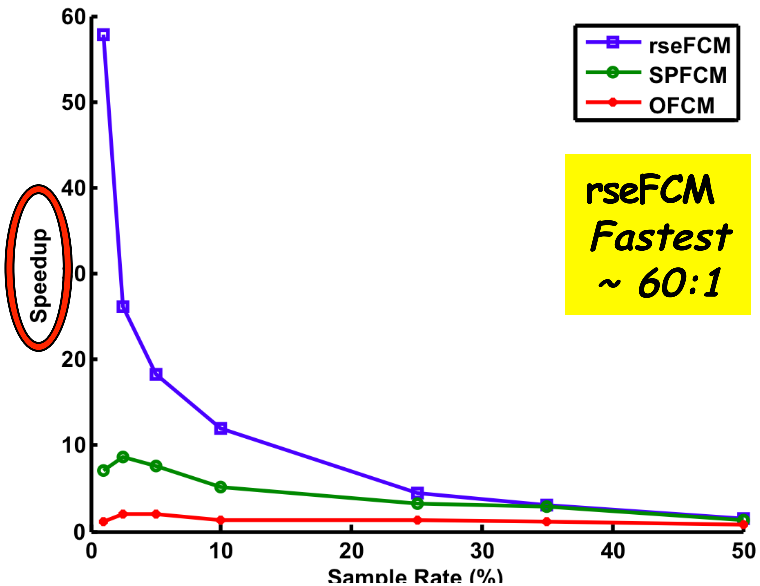


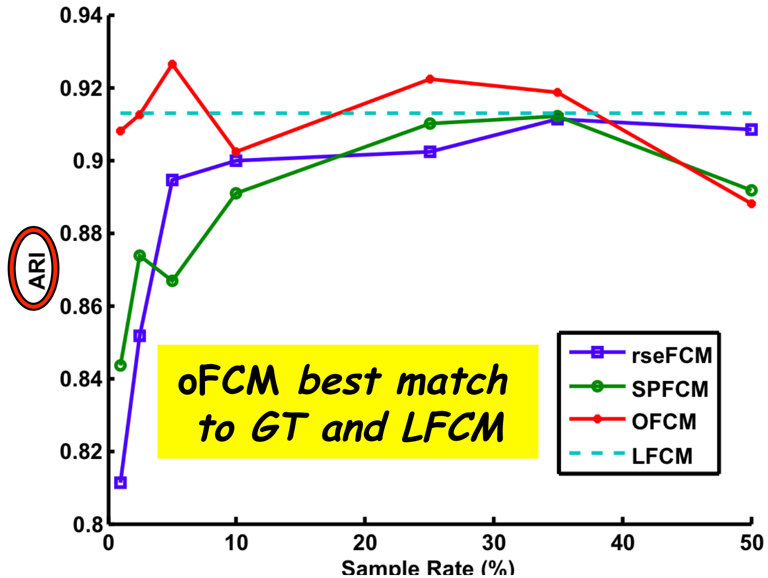

# 2D15 Data

n=5000

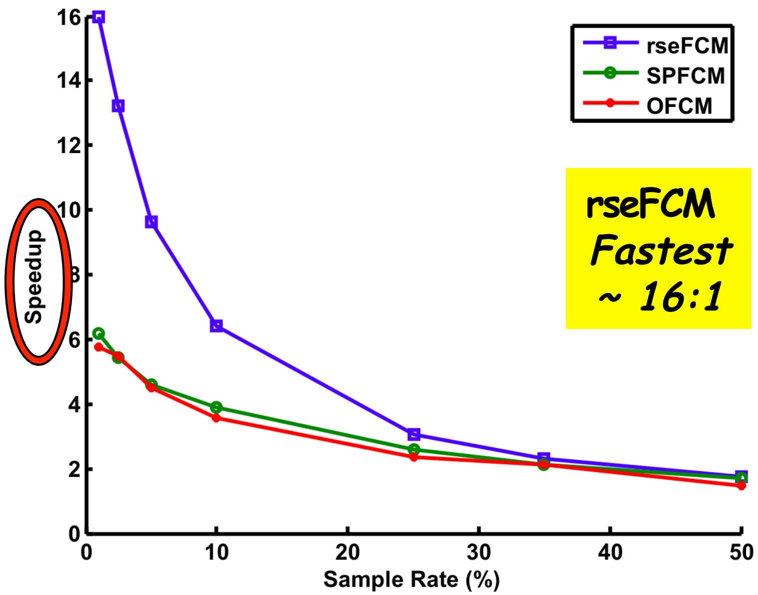
p=2

c=15

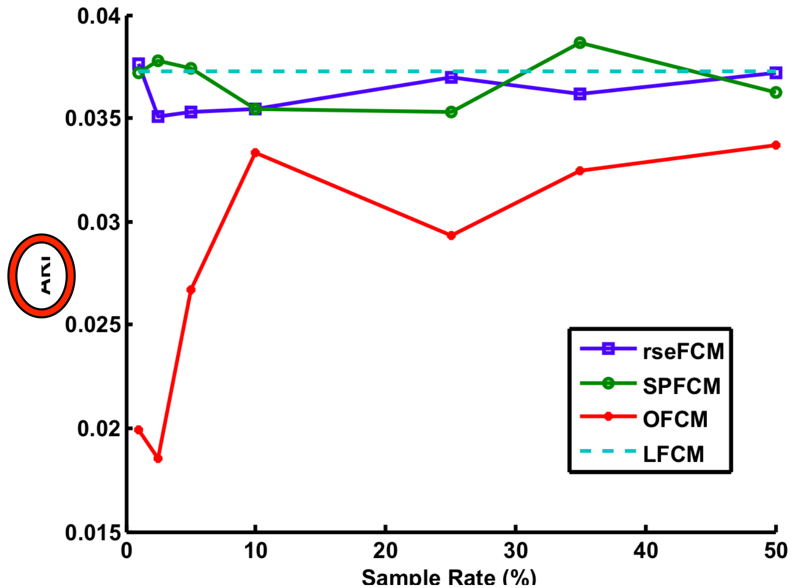


# Forest Data n=581,012, p=54, c=7




rseFCM Fastest ~ 16:1




rseFCM/spFCM best matches to GT and LFCM

# MRI Image Data n ~ $4 \times 10^6$, p=1 or 3, c=3

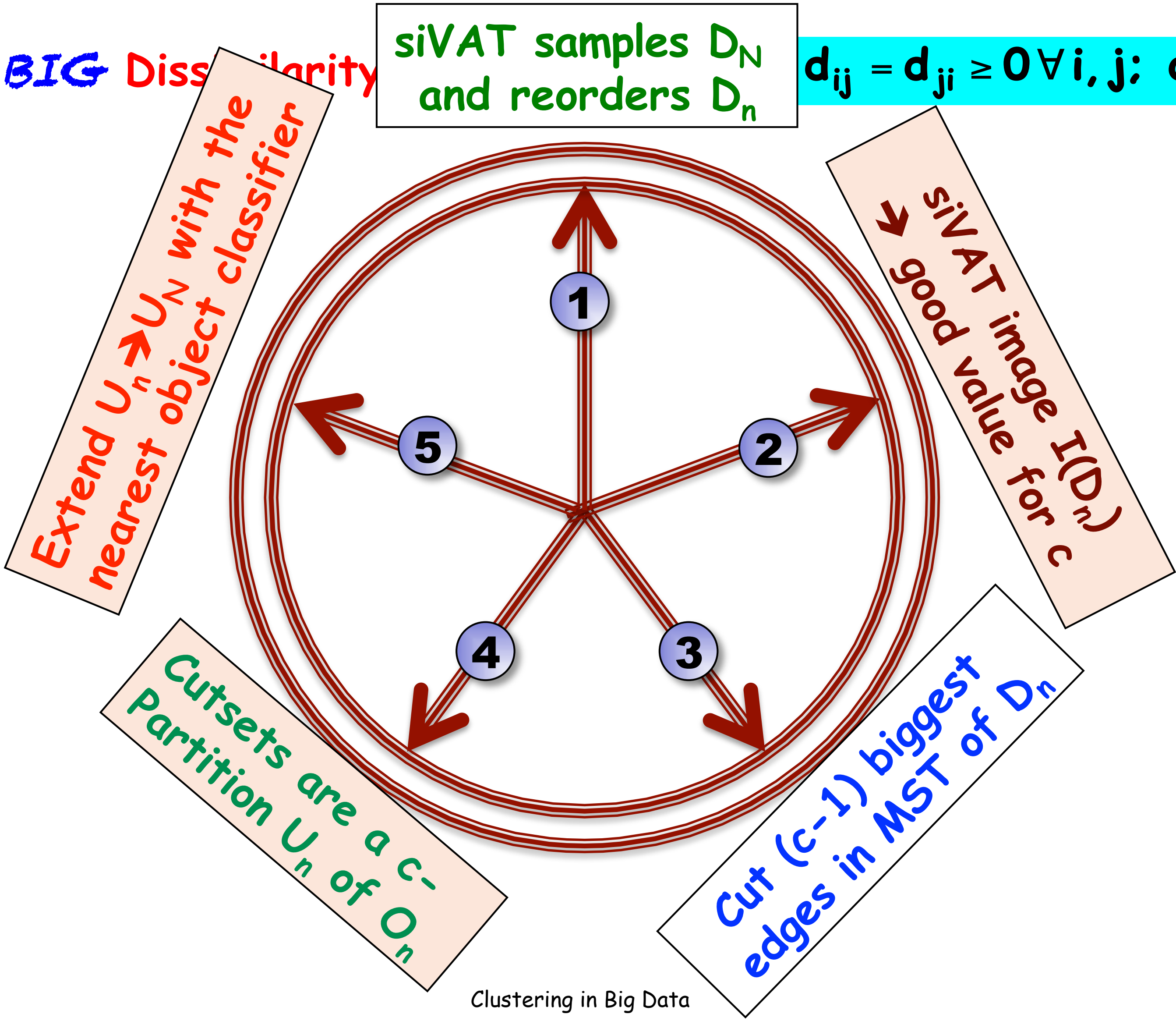
| p=1 | 0.1% samples SU | $ARI_2$ | $ARI_s$ | 1% samples SU | $ARI_2$ | $ARI_s$ | 10% samples SU | $ARI_2$ | $ARI_s$ |
|---|---|---|---|---|---|---|---|---|---|
| rseFCM | 22 | 0.97 | 0.66 | 18 | 0.99 | 0.66 | 7 | 1 | 0.66 |
| spFCM | 13 | 0.98 | 0.66 | 13 | 0.98 | 0.66 | 8 | 0.98 | 0.66 |
| oFCM | 2 | 1 | 0.66 | 4 | 1 | 0.66 | 4 | 1 | 0.66 |
| brFCM | 108 | 1 | 0.66 | 50 | 1 | 0.66 | 8 | 1 | 0.66 |

| p=3 | 0.1% samples SU | $ARI_2$ | $ARI_s$ | 1% samples SU | $ARI_2$ | $ARI_s$ | 10% samples SU | $ARI_2$ | $ARI_s$ |
|---|---|---|---|---|---|---|---|---|---|
| rseFCM | 29 | 0.97 | 0.47 | 24 | 1 | 0.47 | 8 | 1 | 0.47 |
| spFCM | 18 | 0.96 | 0.46 | 13 | 0.96 | 0.46 | 7 | 0.96 | 0.46 |
| oFCM | 2 | 0.78 | 0.38 | 2 | 0.93 | 0.44 | 3 | 1 | 0.47 |

SU = "speed up" : $ARI_2(H(U), H(U_{LFCM}))$: $ARI_s(U, U_{LFCM}))$

# clusiVAT for BIG data

Input | ***BIG*** Dissimilarity | $d_{ij} = d_{ji} \geq 0 \,\forall i,j;\; d_{ii} = 0 \,\forall i$



# sVAT/siVAT with maximin sampling for BIG data

**Input** $D_{N \times N} : D_{ij} \geq 0 \; ; \; D_{ii} = 0 : D = D^T$

**Initialize**

$c' \geq c$ An (OVER) estimate of c

$n \leq 6000$ Approximate sample size

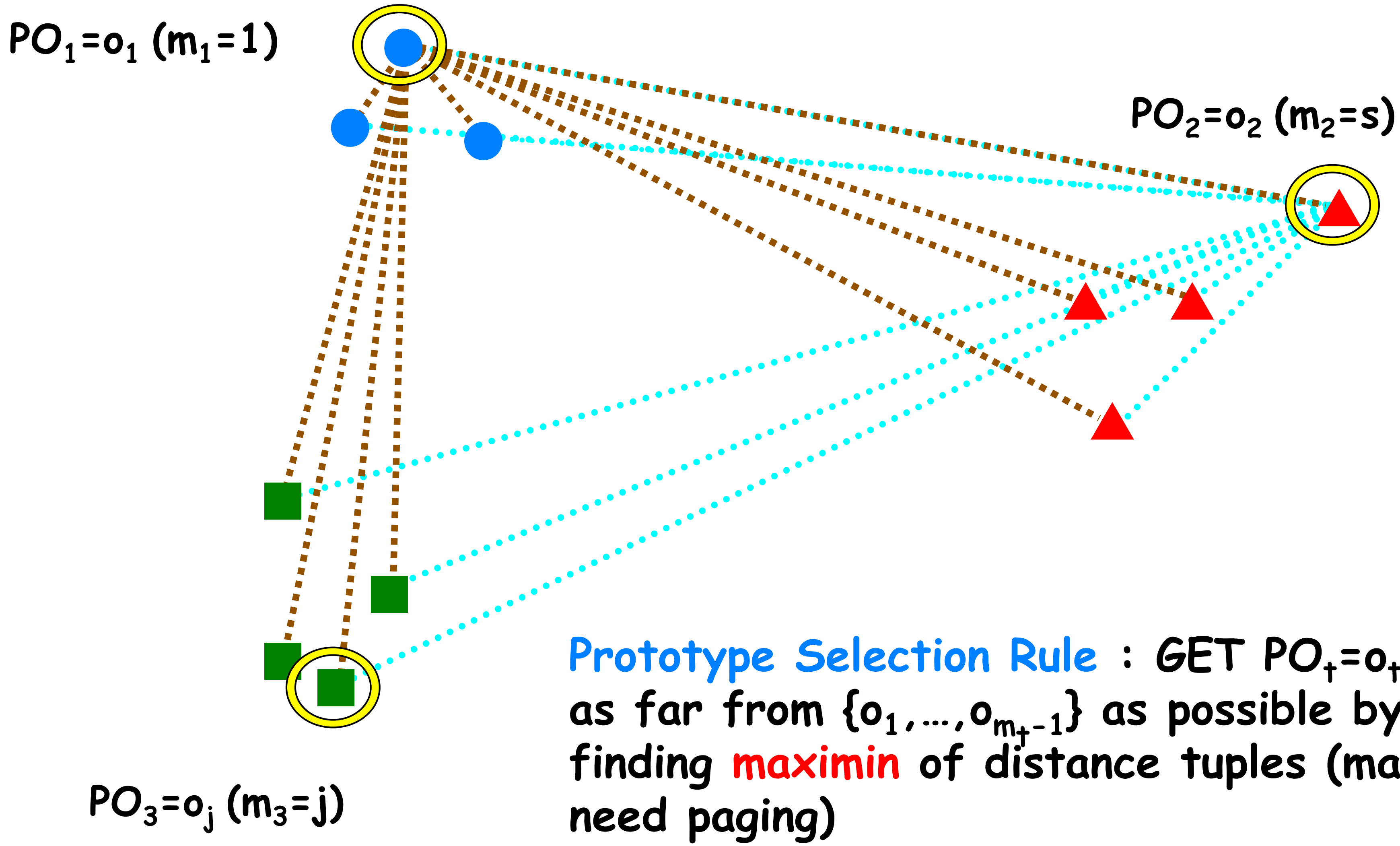
$m_1 = 1$ (*arbitrary*) $o_1$ = 1st Prototype (Index)

$d = (d_1, \ldots d_N) = (D_{11}, \ldots D_{1N})$ search array

**Get c' Maximin Samples; i.e.,**

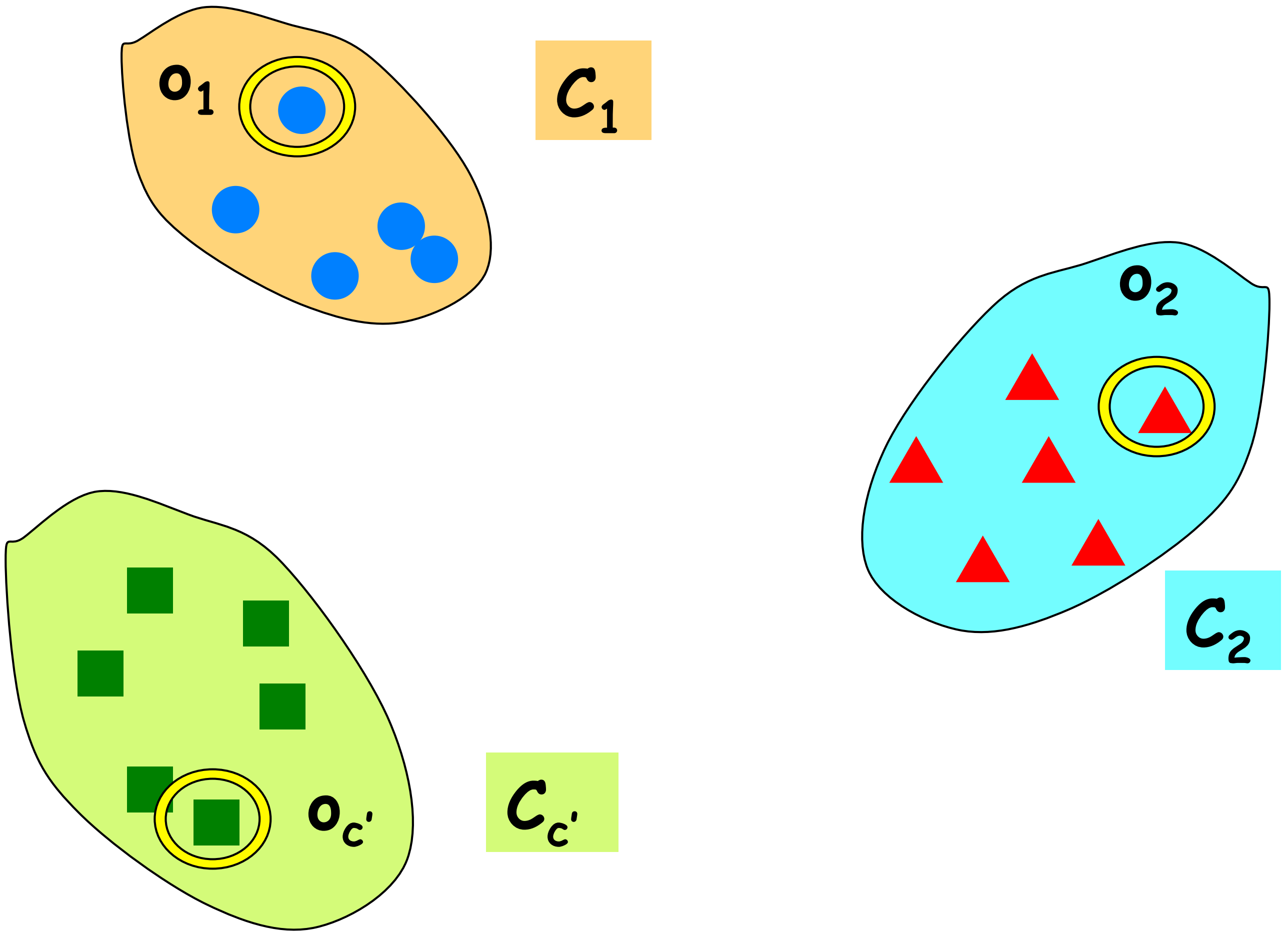
Get indices $\{m_i\}$ of $(c')$ prototypes $= \{o_{m_1}, \ldots, o_{m_k}, \ldots, o_{m_{c'}}\}$

# What are Maximin Samples?

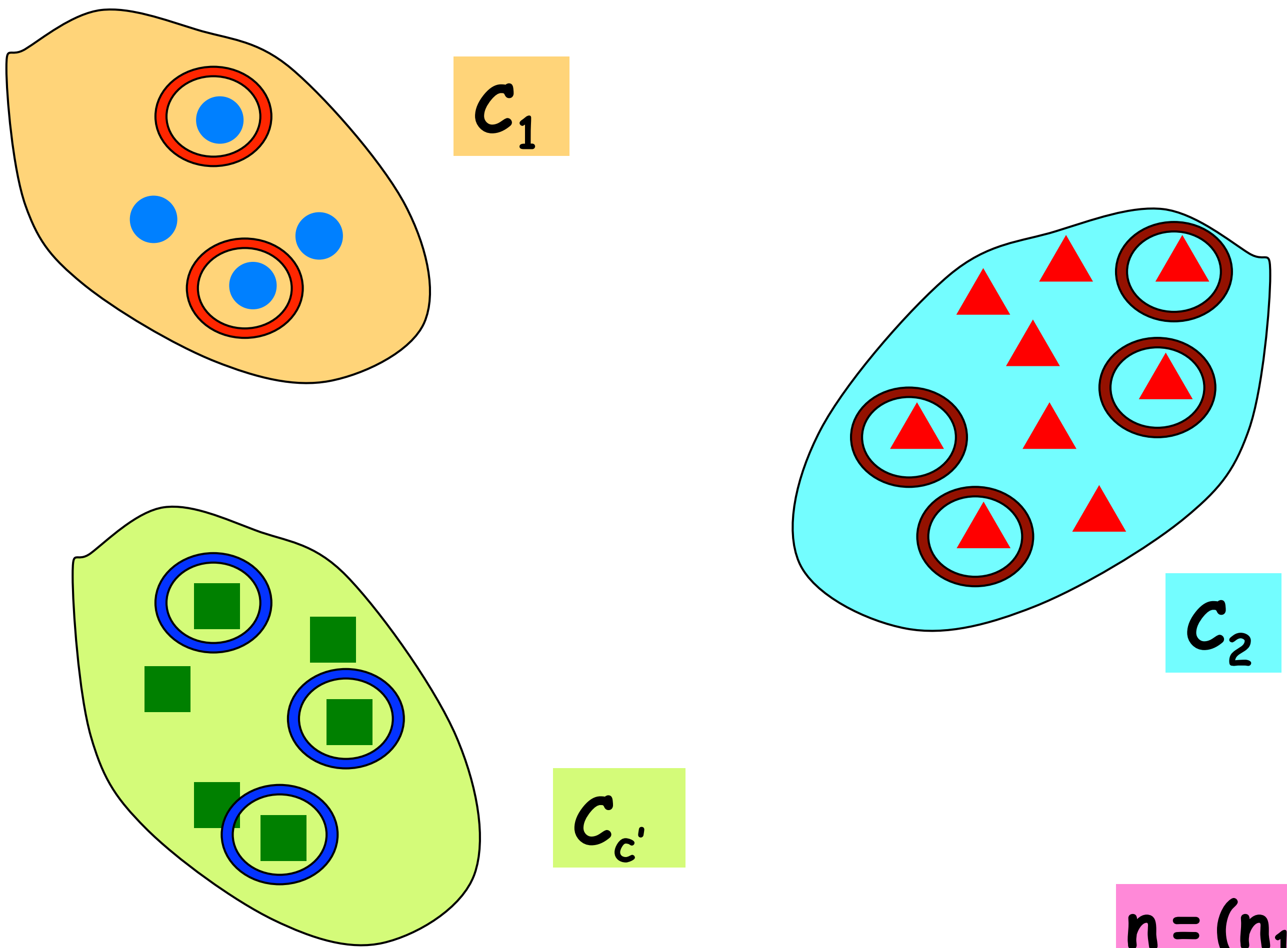



**Prototype Selection Rule** : GET $PO_t=o_t$ as far from $\{o_1,\ldots,o_{m_t-1}\}$ as possible by finding **maximin** of distance tuples (may need paging)

# Get crisp 1-np (HCM) clusters of $\{o_{mk}\}$ - (may need paging)

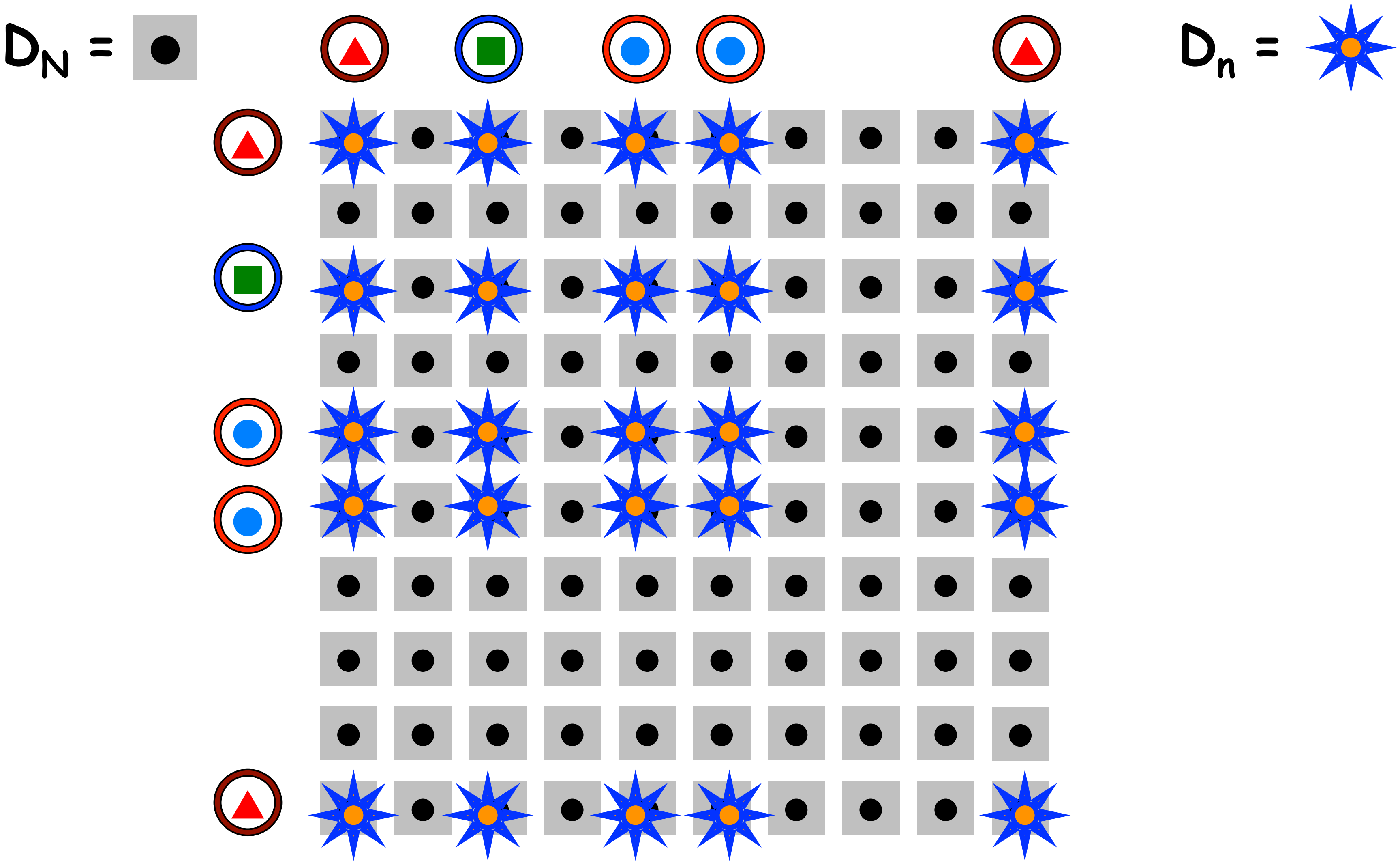


# Randomly Sample cluster $C_t$ $n_t = \left\lceil \frac{n|C_t|}{N} \right\rceil$ times




$$n = (n_1 + \cdots + n_{c'})$$

$D_N =$ $D_n =$

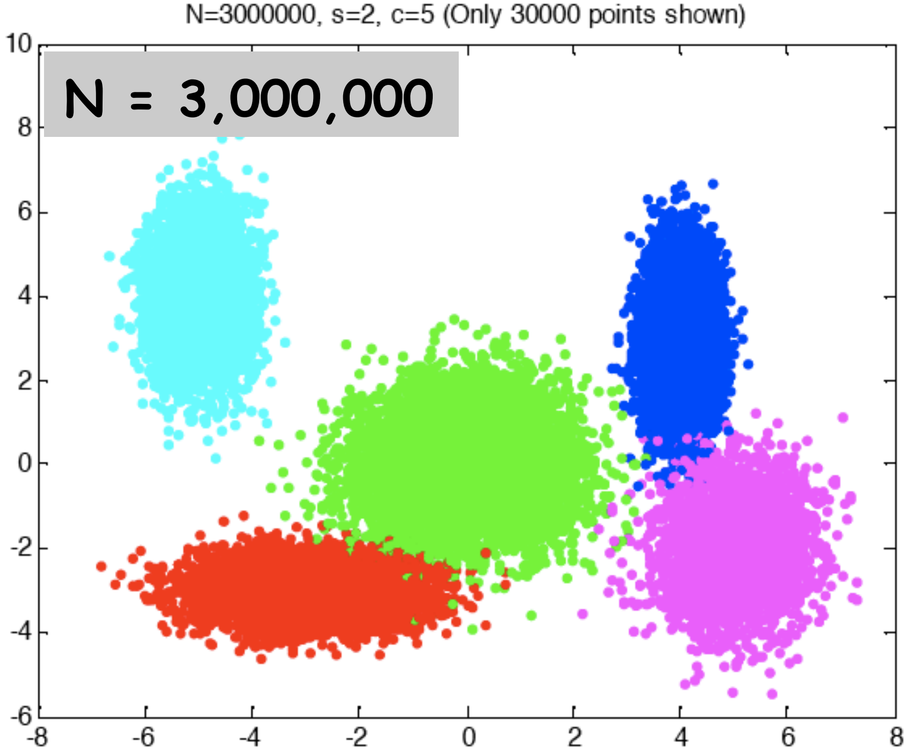


$iVAT(D_n) \approx iVAT(D_N)$ (for visual assessment)

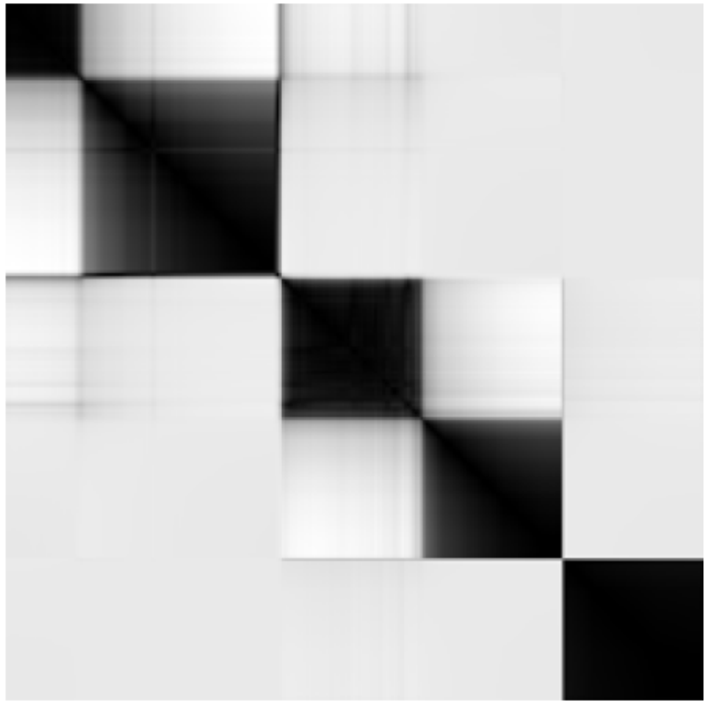
$MST(D_n) \approx MST(D_N)$ (for big data clusiVAT)

# A second approach: *e spec VAT* for VL data

$D_{VL}$=D(X) is unloadable ~ $O(10^{12})$

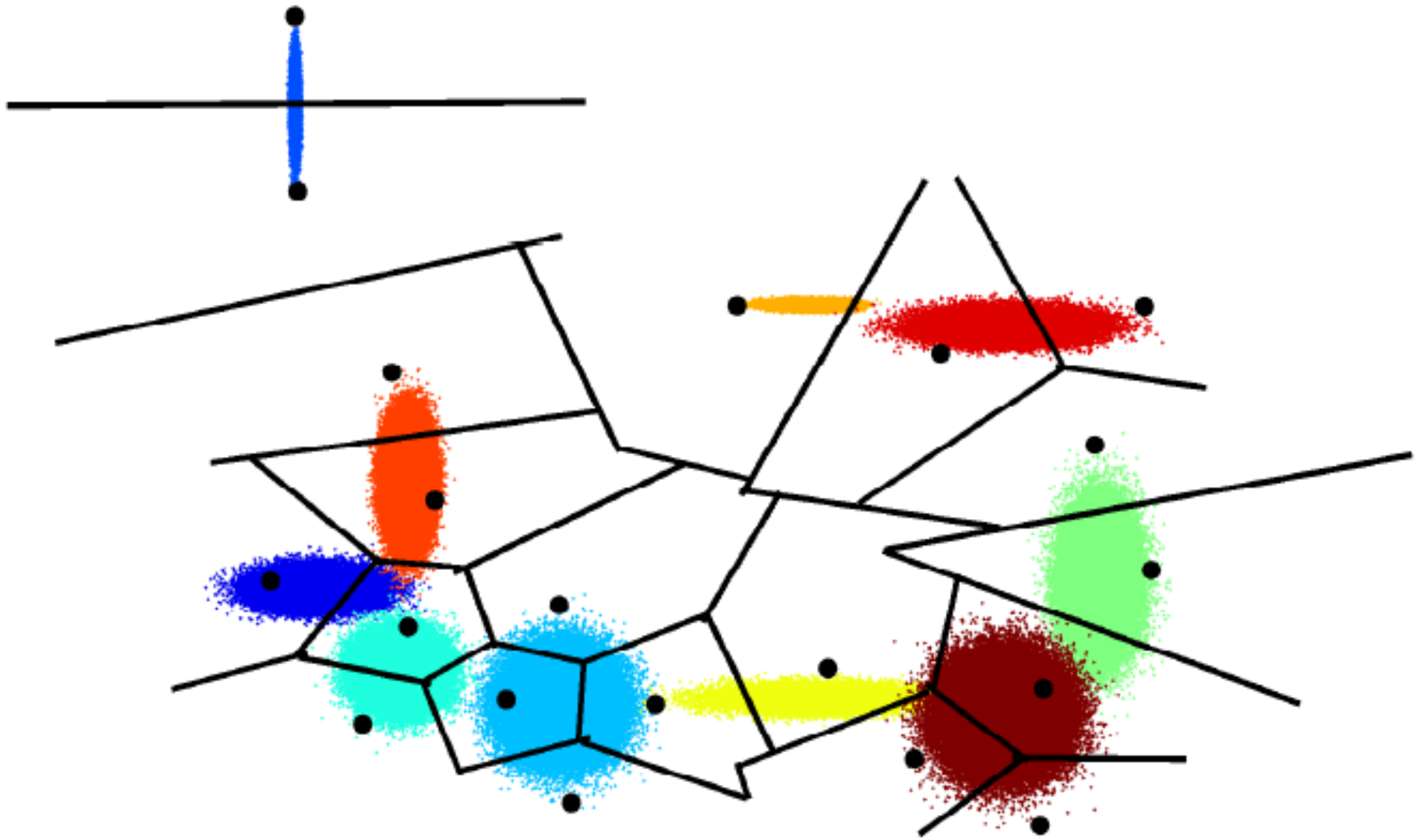



espec iVAT image I(D'*) of n = 2500 samples



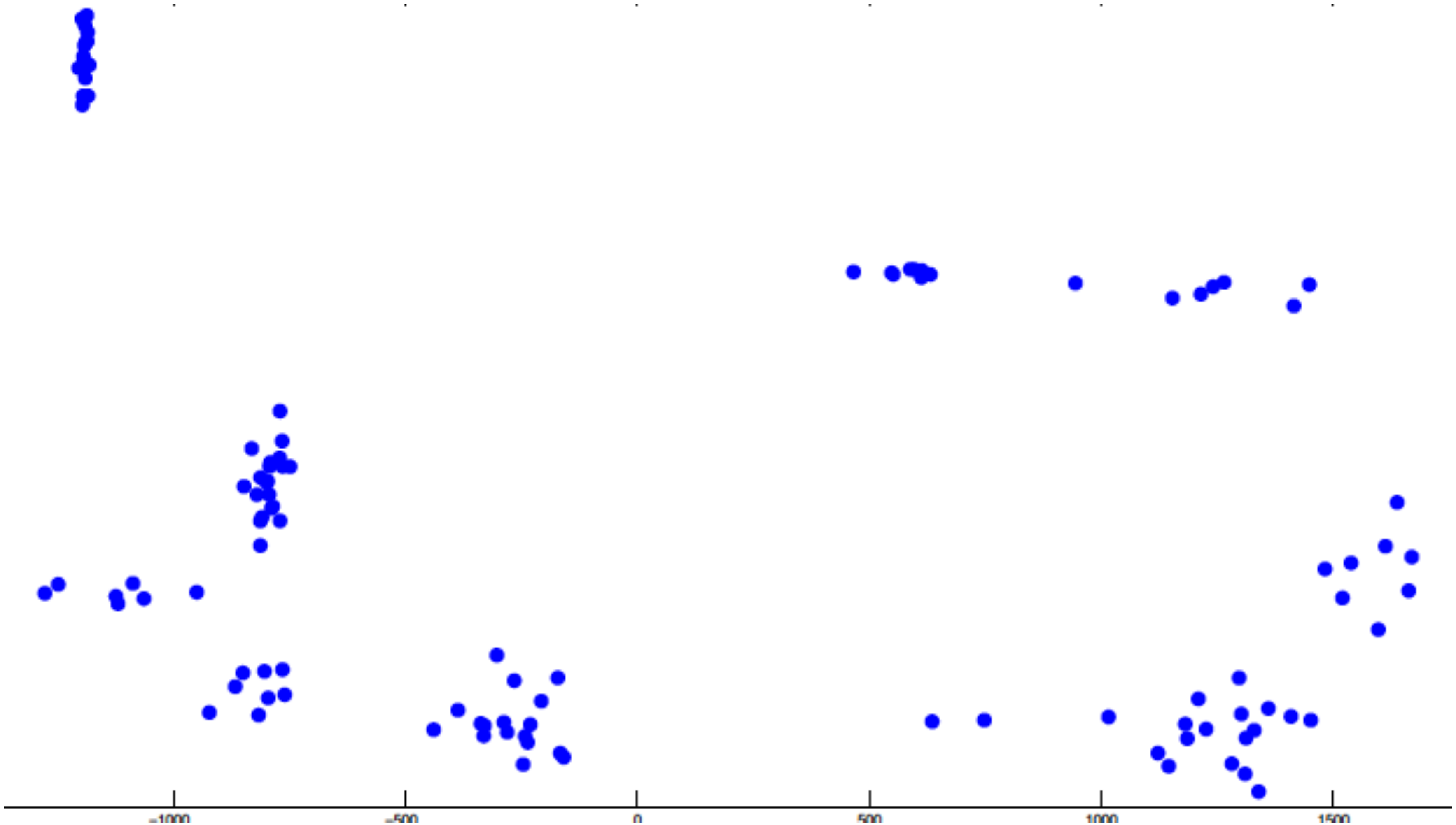
Wang/Geng/Bezdek/Leckie/Kotagiri, (2010). spec-VAT for cluster analysis, *IEEE TKDE*.

## X = Gaussian Clusters: c = 10, N = 1,000,000, p = 2

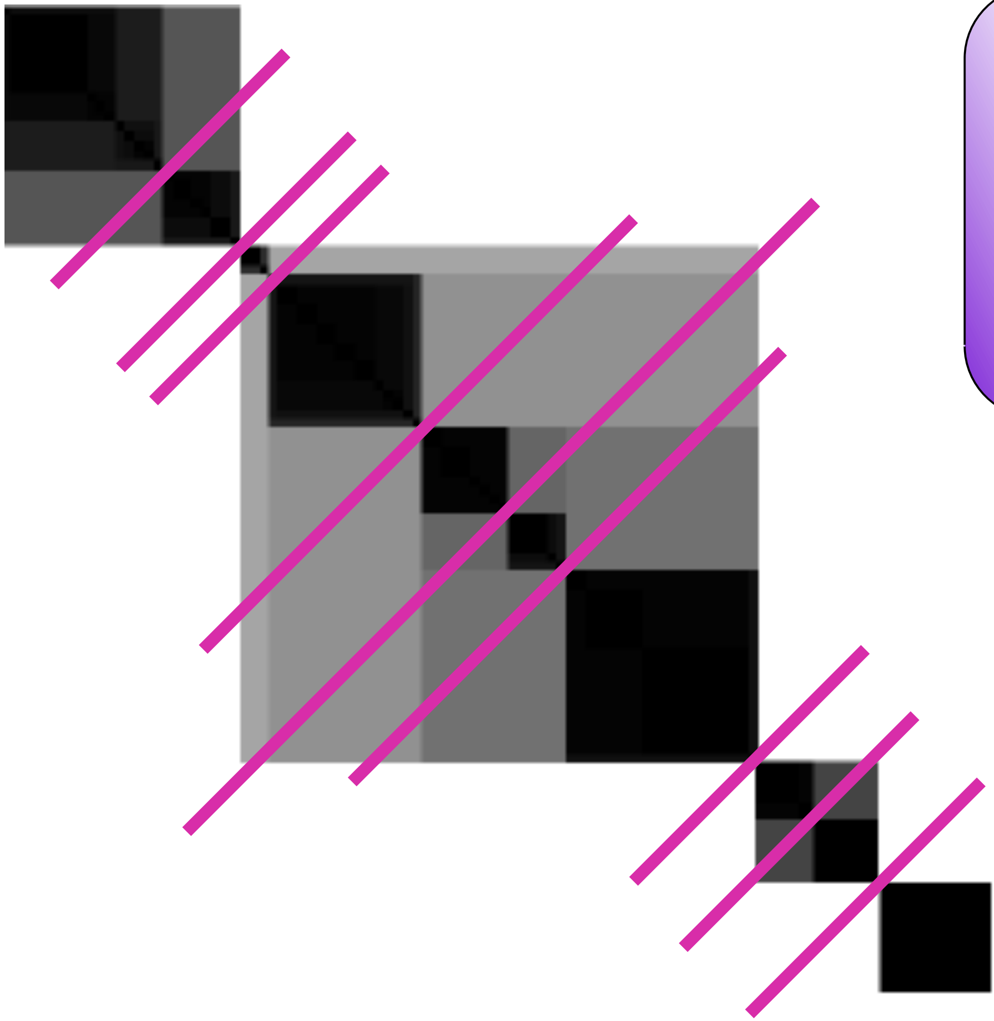


c' = 20 prototypes (•) and 1-np partition (|) of X

# n = 100 random samples from the 20 partitions



# clusiVAT image of the n = 100 samples implies c = 10

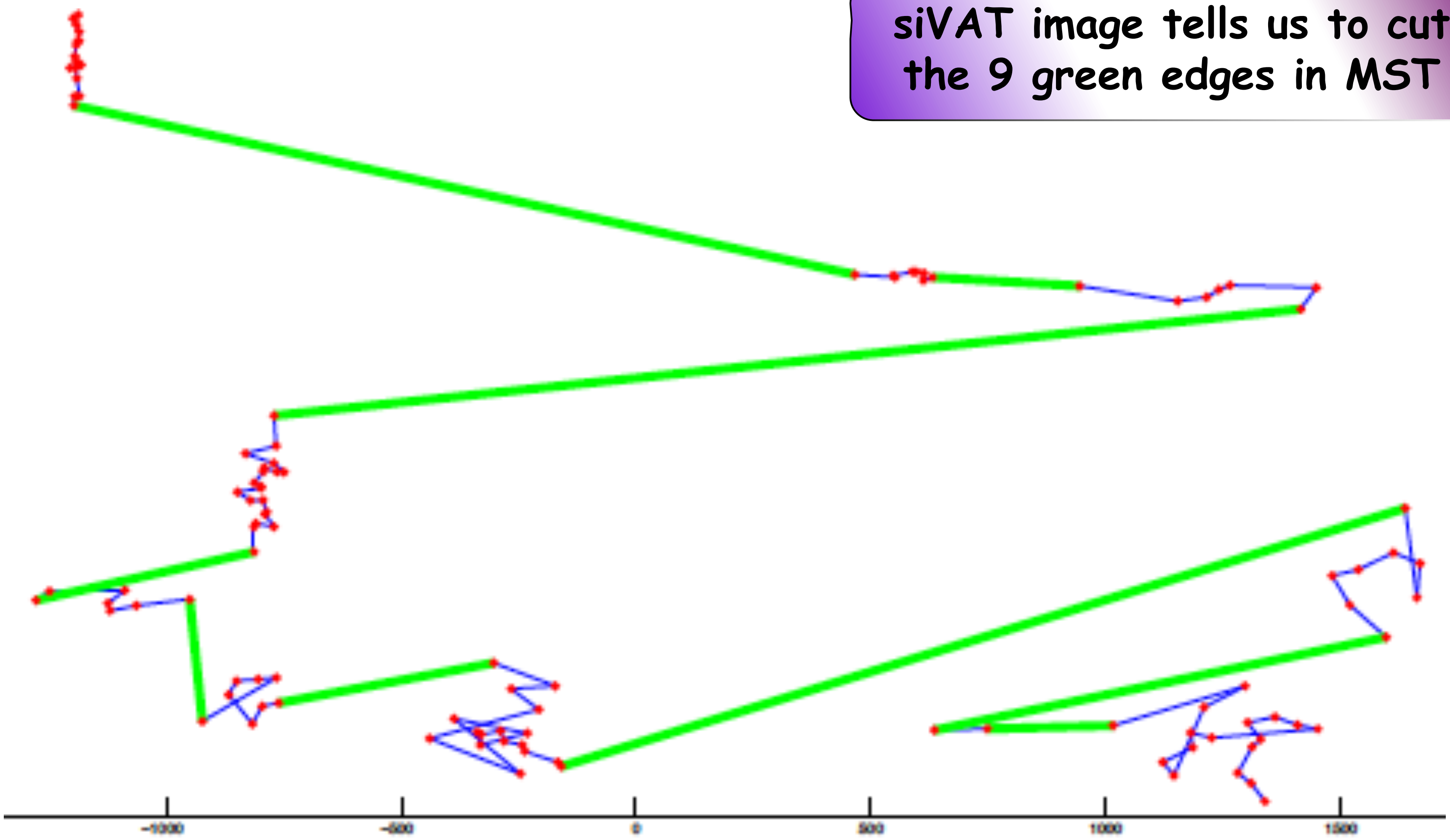


siVAT image suggests that c = 10

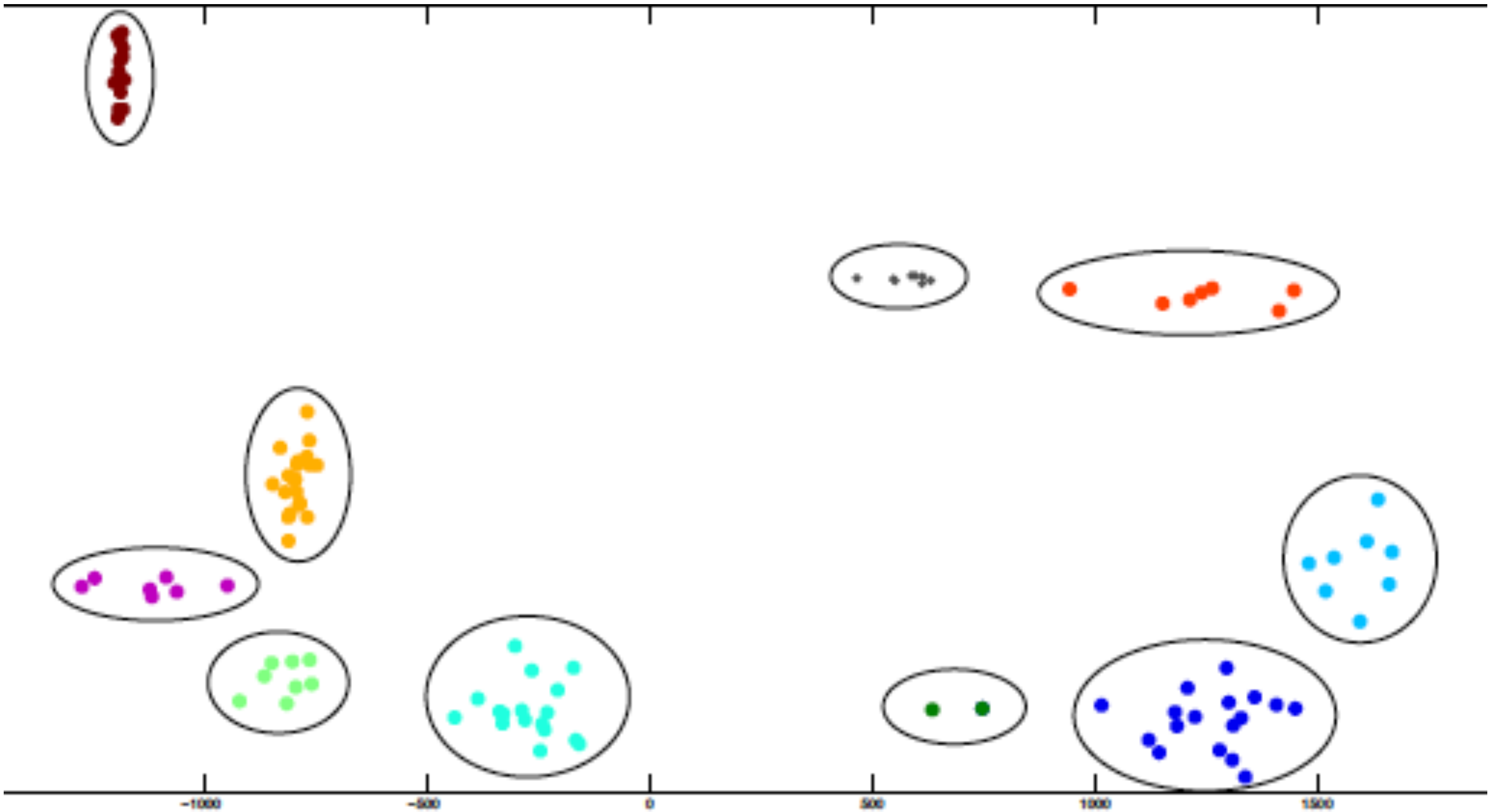
So clusiVAT will cut the 9 largest edges in the MST on $D_n$

# MST on the n = 100 samples

siVAT image tells us to cut the 9 green edges in MST

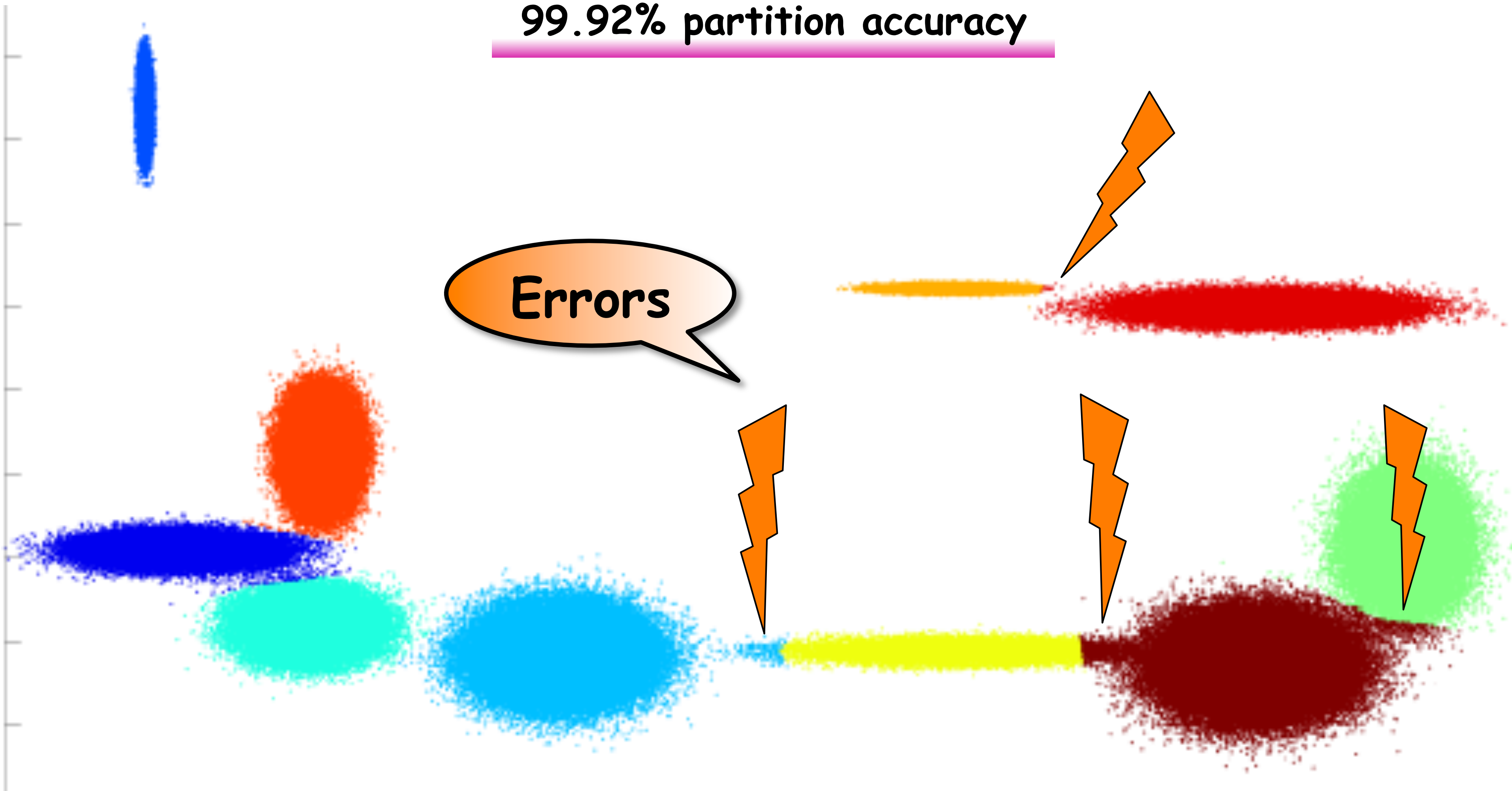


# ClusiVAT 10-partition of the n = 100 samples



## clusiVAT 10-partition of the BIG data: N = 1,000,000

99.92% partition accuracy



# Comparing 5 *crisp* BD clustering algortithms: clusiVAT, HCM="*k-means*", spHCM, oHCM, CURE

Time : CPU time in secs

Partition Accuracy of crisp U

$$PA(U\,|\,U_{GT}) = \frac{\langle U_{GT}, U\rangle}{n} = \frac{\sum_{i=1}^{c} n_i}{n} = \left(\frac{\#\ \text{matched}}{\#\ \text{tried}}\right)$$

$U_{GT}$ = "ground truth" partition of crisp labels

# 25 run averages for 12 small sets of CS Gaussian Clusters

| Data-set Information | | | clusiVAT | | k-means | |
|---|---|---|---|---|---|---|
| Total No. of points | Clusters | DI | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) |
| 1000 | 3 | 1.199 | 100 | 0.030 | 76.264 | 0.013 |
| 1000 | 4 | 1.037 | 100 | 0.059 | 79.310 | 0.023 |
| 1000 | 5 | 1.039 | [illegible] | [illegible] | [illegible] | [illegible] |
| 2000 | 3 | 1.006 | | | | |
| 2000 | 4 | 1.195 | | | | |
| 2000 | 5 | 1.030 | | | | |
| 5000 | 3 | 1.036 | | | | |
| 5000 | 4 | 1.175 | | | | |
| 5000 | 5 | 1.075 | | | | |
| 10,000 | 3 | 1.181 | | | | |
| 10,000 | 4 | 1.121 | | | | |
| 10,000 | 5 | 1.120 | | | | |
| Average values | | | | | | |

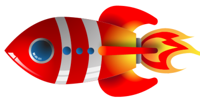
| Single pass k-means | | Online k-means | | CURE | |
|---|---|---|---|---|---|
| Accuracy (%) | Time (s) | Accuracy (%) | Time (s) | Accuracy (%) | Time (s) |
| 85.802 | 0.023 | 70.769 | 0.018 | 100 | 11.236 |
| 99.071 | 0.032 | 62.087 | 0.020 | 100 | 10.925 |
| 81.566 | 0.029 | 55.265 | 0.016 | 100 | 10.722 |
| 100 | 0.022 | 85.835 | 0.021 | 100 | 11.040 |
| 100 | 0.029 | 78.709 | 0.028 | 100 | 10.921 |
| 66.865 | 0.088 | 45.888 | 0.030 | 100 | 10.823 |
| 100 | 0.067 | 84.811 | 0.051 | 100 | 11.344 |
| 100 | 0.053 | 67.472 | 0.057 | 100 | 11.098 |
| 97.850 | 0.062 | 77.291 | 0.062 | 100 | 10.853 |
| 100 | 0.086 | 94.983 | 0.098 | 100 | 11.395 |
| 100 | 0.094 | 73.034 | 0.102 | 100 | 11.284 |
| 100 | 0.094 | 73.901 | 0.108 | 100 | 11.110 |
| 94.263 | 0.057 | 72.504 | 0.051 | 100 | 11.063 |

Mean averages for 12 BIG sets of CS Gaussian Clusters. Ave. Size N = 450,000

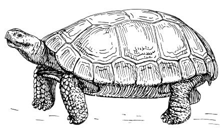
| | c'iVAT | h-km | sp-hkm | ol-hkm | CURE | |
|---|---|---|---|---|---|---|
| large CS | 0.977 | 4.487 | 3.934 | 3.834 | 31.30 | time, secs |
| | 100 | 72.77 | 99.67 | 70.19 | 99.81 | accuracy,% |

Mean averages for 12 BIG sets of non-CS Gaussian Clusters. Ave. Size N = 450,000

| | c'iVAT | h-km | sp-hkm | ol-hkm | CURE | |
|---|---|---|---|---|---|---|
| large NCS | 1.021 | 4.395 | 5.163 | 4.680 | 31.04 | time, secs |
| | 99.99 | 75.14 | 90.99 | 73.93 | 97.83 | accuracy,% |



clusiVAT is fastest AND most accurate



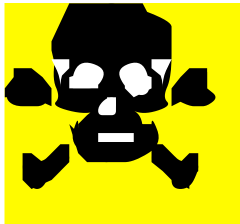
CURE ~ 30 times slower; 2nd best accuracy

**Forest Data** N = 581,012, p=54, c=7 (labeled classes)

| 10 continuous features | 40 binary soil types | 4 binary wilderness types |
|---|---|---|

siVAT image on n = 70
Forest samples

k = 7 clusters ?
Probably k ≥ 20



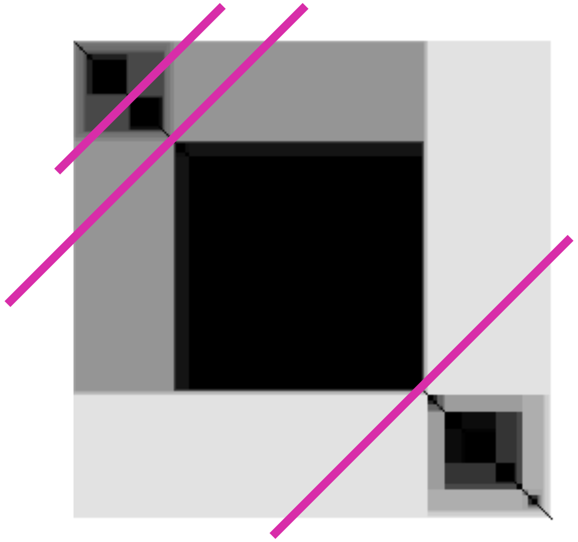
| | c'iVAT | h-km | sp-hkm | ol-hkm | CURE | |
|---|---|---|---|---|---|---|
| Forest | 4.049 | 46.53 | 59.24 | 173.5 | 59.48 | ⇒ time, secs |
| | 43.7 | 11 | 15 | 4 | 43.6 | ⇒ accuracy, % |

KDD-99 Cup data: (22 simulated attacks + normal data) ➔ c = 23

41 features in [0, 1]

N = 4,292,637

c = 23 class labels



siVAT image for n = 230

4 (major) attack types

Denial of Service (DOS)
Users to Root (U2R)
Remote to Local (R2L)
Probing Attacks (PROBE)

| c'iVAT | hkm | spkm | olkm | CURE | |
|---|---|---|---|---|---|
| 97.06 | 94.25 | 96.45 | 94.87 | 91.54 | ⇨ accuracy,% |
| 76.0 | 124.8 | 120.4 | 138.5 | 841.6 | ⇨ time, secs |

# A *few* acceleration schemes for *literal algorithm* $\mathcal{A}$

| Ref. | $\mathcal{A}$ | p | c | n | speedup |
|---|---|---|---|---|---|
| Arthur | *HCM* | 5-35 | 5-50 | 10,000,0.5M | 1-9.6:1 |
| Hore | *HCM* | 3-617 | 3-12 | 150, 4M | 600,000:1 |
| Pelleg | *HCM* | 2 | 5000 | 0.4M | 26-136:1 |
| Moore | *EM* | 2-6 | 5,320 | 12,500,0.7M | 9-500:1 |
| Thiesson | *EM* | 2, 33 | 303,12 | 21,888,0.6M | 1.7-2.8:1 |
| Ortiz | *EM* | 2 | 2 | 2,000 | 1-12:1 |
| March | *SL* | 3,3840 | 10, v | $4(10^4)-10^6$ | 3:1 |
| Müllner | *SL* | 2, 100 | 1, 5 | 10,10,000 | 10:1 |

# A *few* acceleration schemes for $\mathcal{A}$ = *fuzzy c-means*

| Ref. | $\mathcal{A}$ | p | c | n | speedup |
|---|---|---|---|---|---|
| Cannon | $FCM$ | 10 | 10 | 0.25 mb | 6:1 |
| Kamel | $FCM$ | v | v | Small | 1.2 : 1 |
| Cheng | $FCM$ | 3, 6 | 10 | 0.4 mb | 3:1 |
| Altman | $FCM$ | 3 | 3 | 1 mb | 3-10:1 |
| Kolen | $FCM$ | 9 | 10 | 20 mb | 9:1 |
| Borgelt | $FCM$ | 8-13 | 2, 3 | $\leq$ 4177 | 2:1 |
| Anderson | $FCM$ | 4-32 | 4-64 | 64,8192 | 10-100:1 |
| Eschrich | $FCM$ | 2,3 | 5,7 | 0.4mb | 59-290:1 |

# Empirical Conclusions: pseFCM & rseFCM

**Sampling** ⇨ Three types (random, progressive, Maximin). Easily adaptable for extensions to Big Data with *many* other algorithms

**Extension** ⇨ Non-iterative scaling for *many* algorithms typically incurs about 1% of total CPU time

**rseFCM** ⇨
- Superiority to pseFCM increases with n
- Faster than spFCM/oFCM for large n
- Average speedup of LFCM ~ 30:1
- Good Approximation to LFCM clusters

# Empirical Conclusions: brFCM, spFCM & oFCM

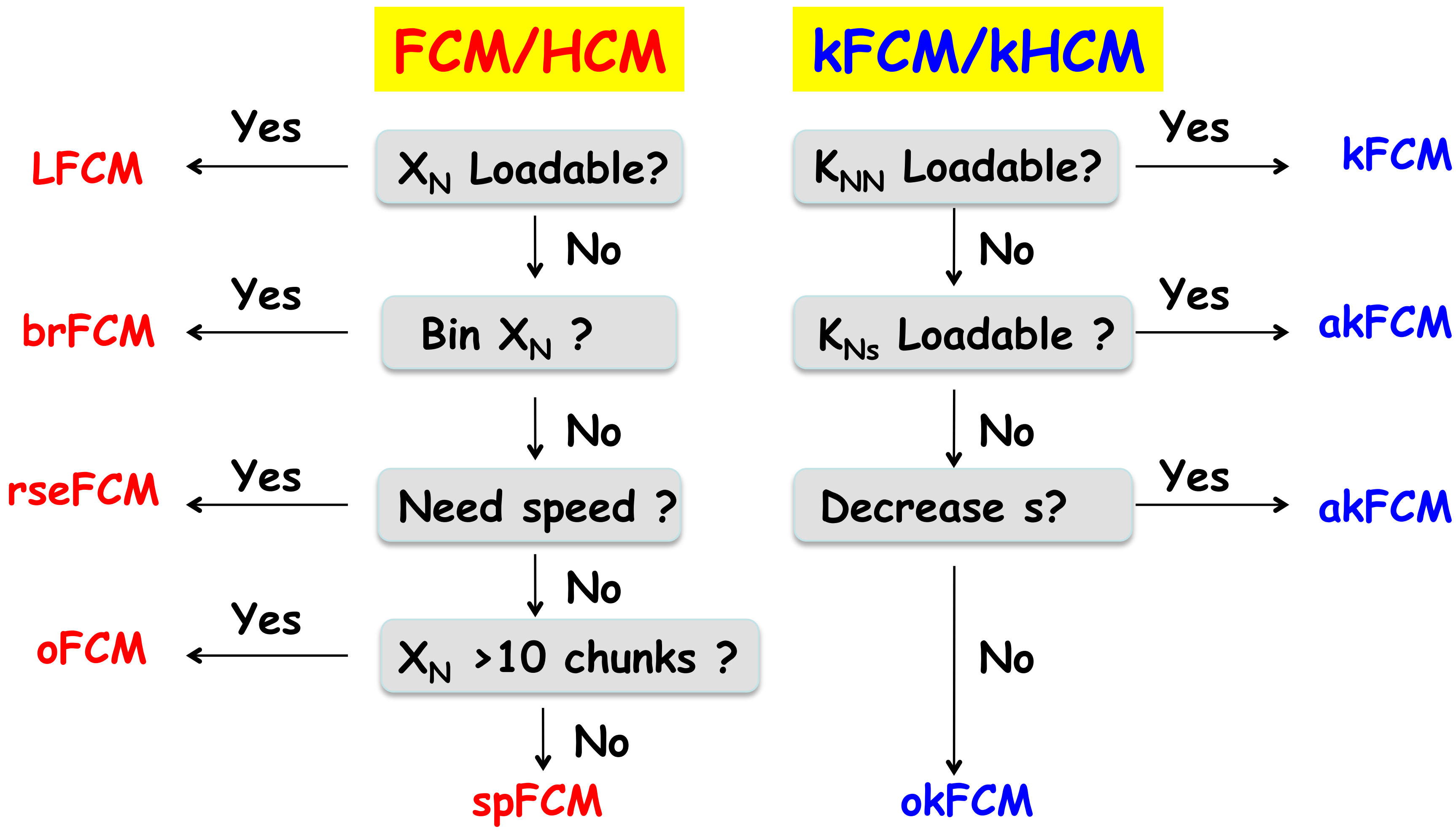
**brFCM** ⇨ Excellent acceleration for 1D images

Average speedup of brFCM ~ 100:1

**spFCM** ⇨ Retains "history" of clusters as more data chunks are added to processing

**oFCM** ⇨ No history retention; useful for on-line streaming analysis (of chunks)

# *Recommendations:* Big Data fuzzy c-Means *AND* its special case, HCM = "k-means" at m=1



# Empirical Conclusions: siVAT and clusiVAT

## ClusiVAT works (so far !)

- the siVAT image usefully estimates c *before* clustering
- is *EXACT* (scalable) SL when DI > 1
- is *much more* accurate than batch and incremental k-means
- is 25-250 times *faster* than CURE

## Things to fix and do

- SL can go awry if data is very "stringy"
- Next up: incremental clusiVAT for streaming data !

# What Happens Next?

**"Data-driven decisionmaking is another sign that the role of the campaign pros in Washington who make decisions on hunches and experience is rapidly dwindling, being replaced by quants and computer coders who can crack massive data sets for insight. As one official put it, the time of "guys sitting in a back room smoking cigars, saying 'We always buy *60 Minutes*'" is over. In politics, *the era of big data has arrived*."**

M. Scherer, Inside the Secret World of Quants and Data Crunchers who helped ObamaWin, *Time Magazine*, Nov. 19, 2012, 56-60.

# This has 2 Results

1

2 ONSLAUGHT OF BIG DATA BUZZWORDS

[7 Vs: volume, velocity, veracity, value, variety, validity, value !!! ]


25/03/2012
OK Grandpa … Time to wrap it up
Grandma is laughing at you !

# WAKE UP


# IT'S OVER


Thanks !
G'Day !

## With these aids my hearing is about 8% of normal

**I will try to answer questions, but a better result follows if you email them to me.**

**Questions, pdf's of today's talk and papers**

**jcbezdek@gmail.com**