Application of the Cross-Entropy Method to Clustering and Vector Quantization

Dirk P. Kroese[†] and Reuven Y. Rubinstein[∘] and Thomas Taimre[†⋄]

[∘] Faculty of Industrial Engineering and Management, Technion, Haifa, Israel

[†]Department of Mathematics, The University of Queensland, Brisbane 4072, Australia

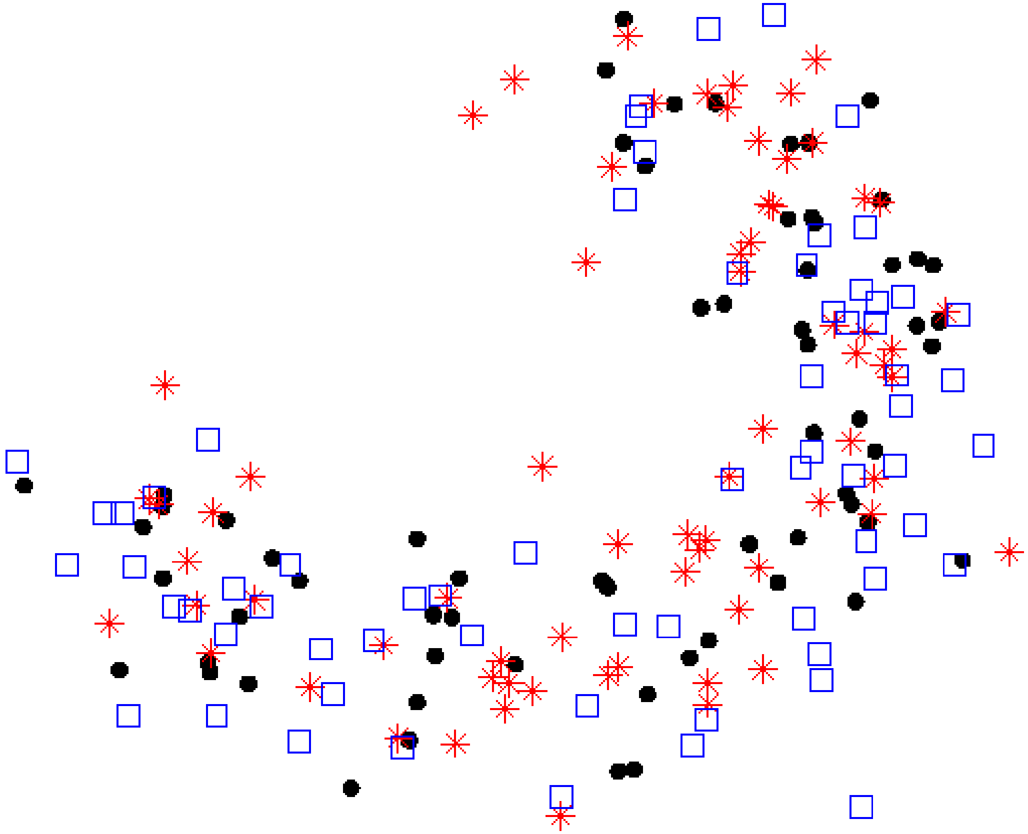
[⋄]Supported by the ARC Centre of Excellence: Mathematics and Statistics of Complex Systems (MASCOS). Presenting Author.

Contents

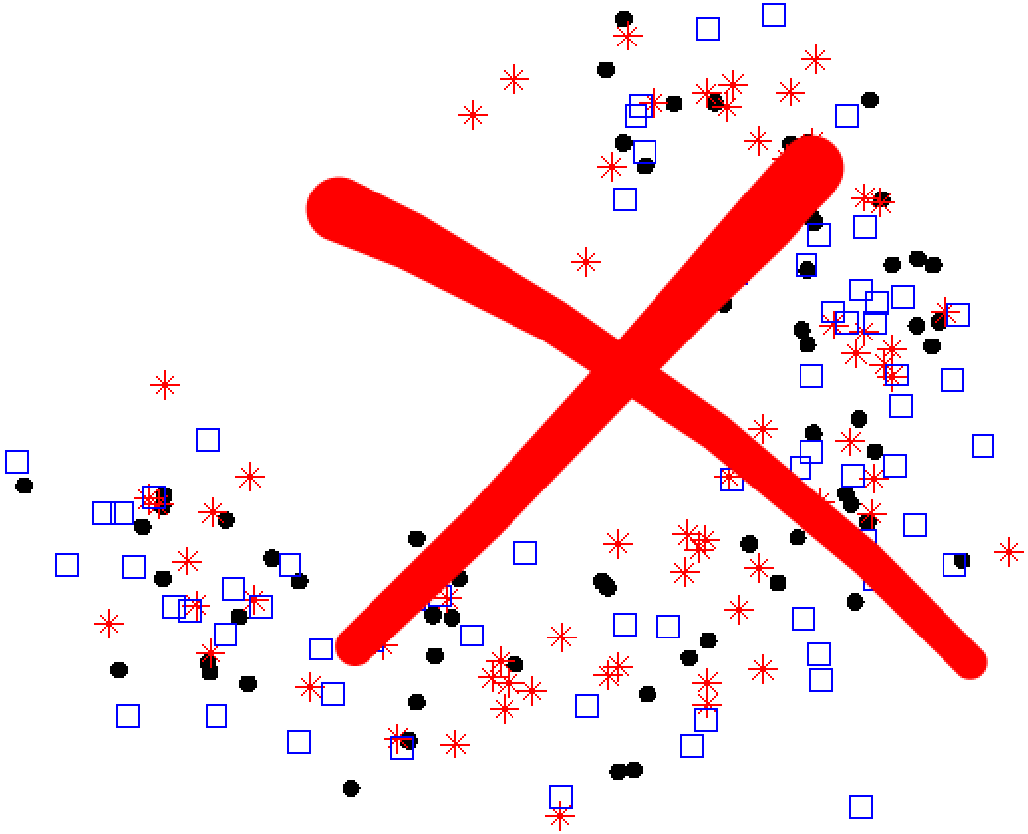
- Introduction
- Two CE Approaches
- Numerical Results
- Application
- Conclusions

The Problem

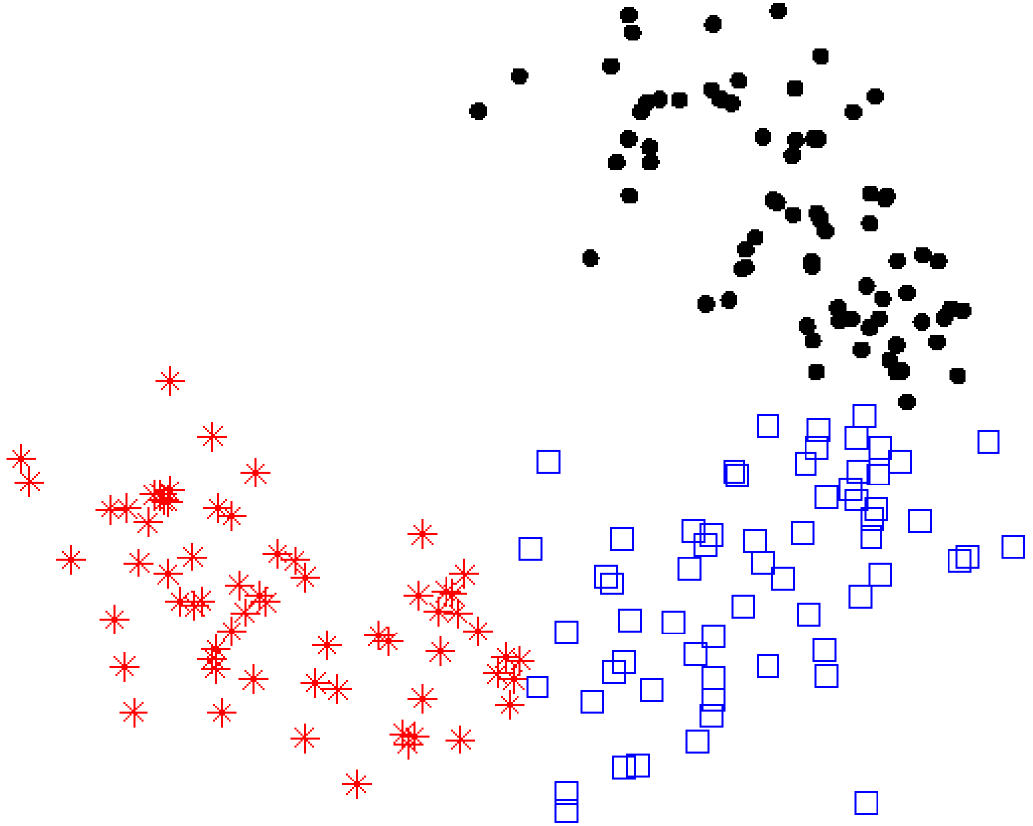
The Problem



The Problem



The Problem



Introduction

- Cluster : Data that are “similar” to each other
- Clustering and vector quantization : How to group “feature” vectors into clusters
- Applications : Communication, data compression and storage, database searching, pattern matching, and object recognition
- What We Do : Apply the cross-entropy (CE) method to problems in clustering and vector quantization.

The Problem in Mathematical Terms

- Dataset $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_n\} \in \mathbb{R}^d$
- Partition into K disjoint clusters, $\{R_j\}$, in order to minimize a loss function:

$$\sum_{j=1}^{K} \sum_{\mathbf{z} \in R_j} ||\mathbf{z} - \mathbf{c}_j||^2$$

$\mathbf{c}_j$ represents the cluster center or *centroid* of cluster R_j.

- Objective : Find $(\mathbf{c}_1, \ldots, \mathbf{c}_K)$ and corresponding partition $\{R_1, \ldots, R_K\}$ that *minimizes* the loss function.
- Two ways to think about this ...

Minimizing The Loss Function

- **Approach 1** : Optimizing Cluster Vector

- **Approach 2** : Optimizing Cluster Centroids

Optimizing Cluster Vector

View loss function as a function of the clusters, rather than the centroids

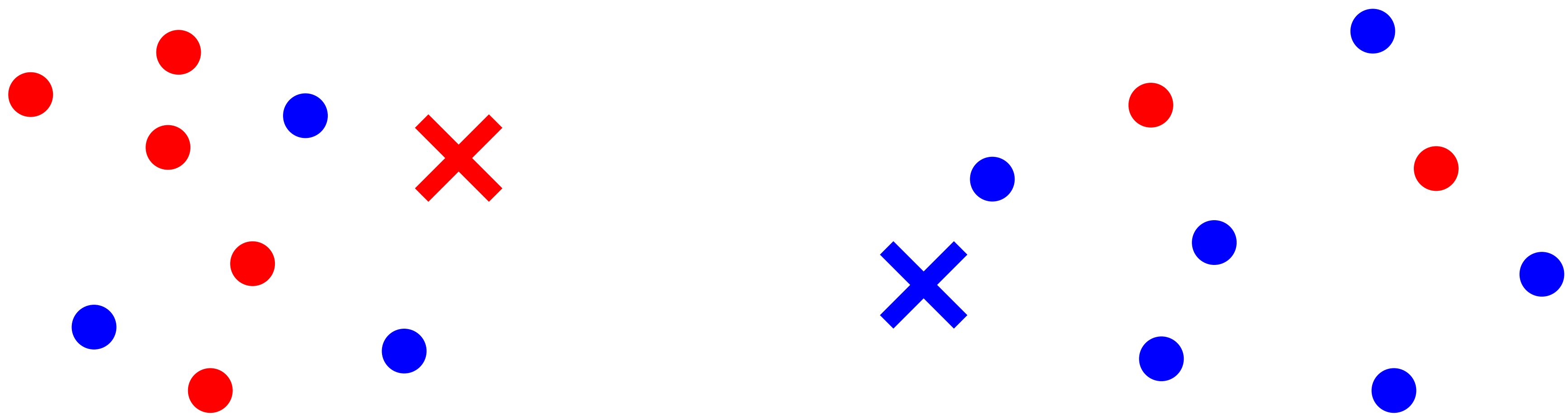
Approach 1

Optimizing Cluster Vector

View loss function as a function of the clusters, rather than the centroids

Optimizing Cluster Vector

View loss function as a function of the clusters, rather than the centroids

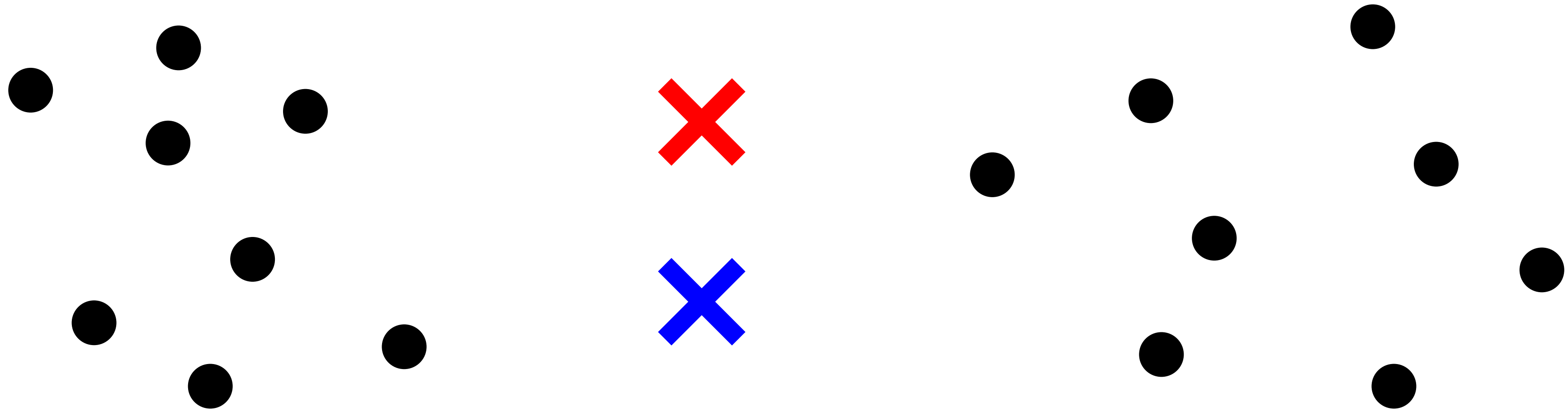


Optimizing Cluster Centroids

View loss function as a function of the centroids, rather than the clusters, with each point assigned to the nearest centroid

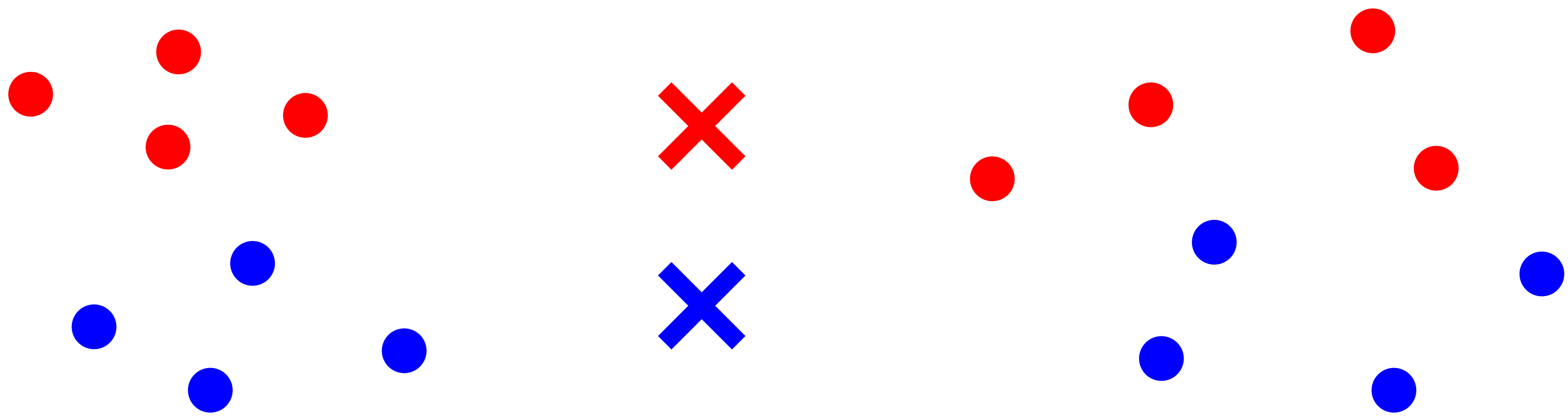
Optimizing Cluster Centroids

View loss function as a function of the centroids, rather than the clusters, with each point assigned to the nearest centroid



Optimizing Cluster Centroids

View loss function as a function of the centroids, rather than the clusters, with each point assigned to the nearest centroid



Generic CE Algorithm

Problem: Maximize performance function $S(\mathbf{x})$ over all states $\mathbf{x}$ in some set $\mathcal{X}$.

Generic CE Algorithm

Problem: Maximize performance function $S(\mathbf{x})$ over all states $\mathbf{x}$ in some set $\mathcal{X}$.

Algorithm:

1. Initialize Set initial parameter vector $\mathbf{v}_0$, set counter $t = 1$
2. Generate Draw $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from pdf with parameter vector $\mathbf{v}_{t-1}$
3. Score Evaluate performance function $S(\mathbf{x})$ for each $\mathbf{X}_i$
4. Update Calculate $\mathbf{v}_t$ via CE using MLE's of the parameters, based on the best-scoring (elite) samples
5. Stop? Stop if convergence to a solution is suspected; otherwise, set $t = t + 1$ and reiterate from Step 2

CE Can Be Applied To Both

- First Approach – Reduce the clustering problem to a combinatorial partition problem with n nodes and K partitions
  - Looking for cluster vector, which assigns a number $1, \ldots, K$ to each point corresponding to the partition $\{R_1, \ldots, R_K\}$
- Second Approach – View optimizing loss function as a continuous multi-extremal optimization problem
  - Looking for centroids, $(\mathbf{c}_1, \ldots, \mathbf{c}_K)$

Approach 1

CE with Cluster Vector

Performance function:

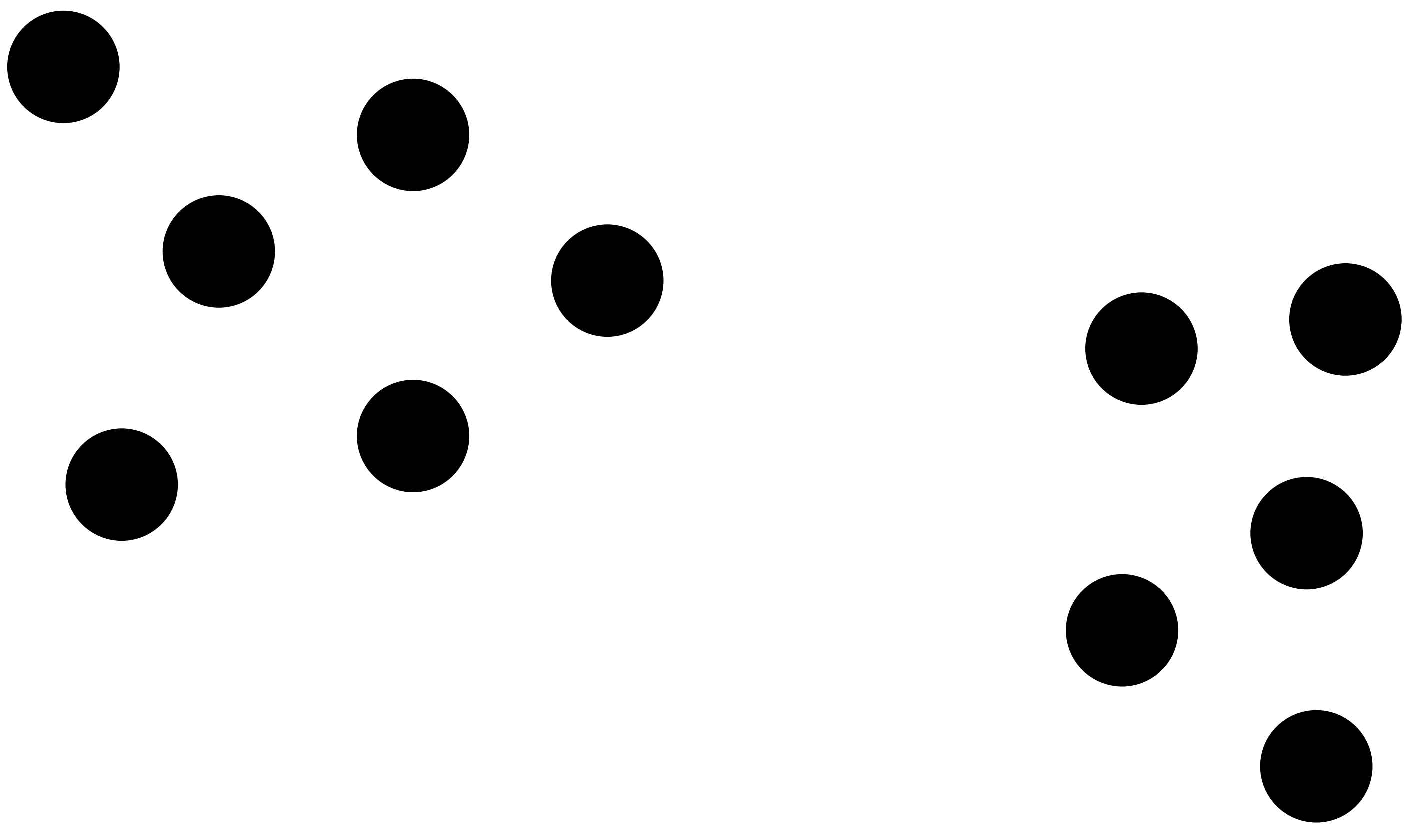
$S(\mathbf{x}) = \sum_{j=1}^{K} \sum_{\mathbf{z} \in R_j} ||\mathbf{z} - \mathbf{c}_j||^2$

1. Draw random cluster vectors $\mathbf{X} \in \mathcal{X}$ from n-dim. discrete distribution with independent marginals, such that $\mathbb{P}(X_i = j) = p_{ij}$, $i = 1, \ldots, n,\ j = 1, \ldots, K$

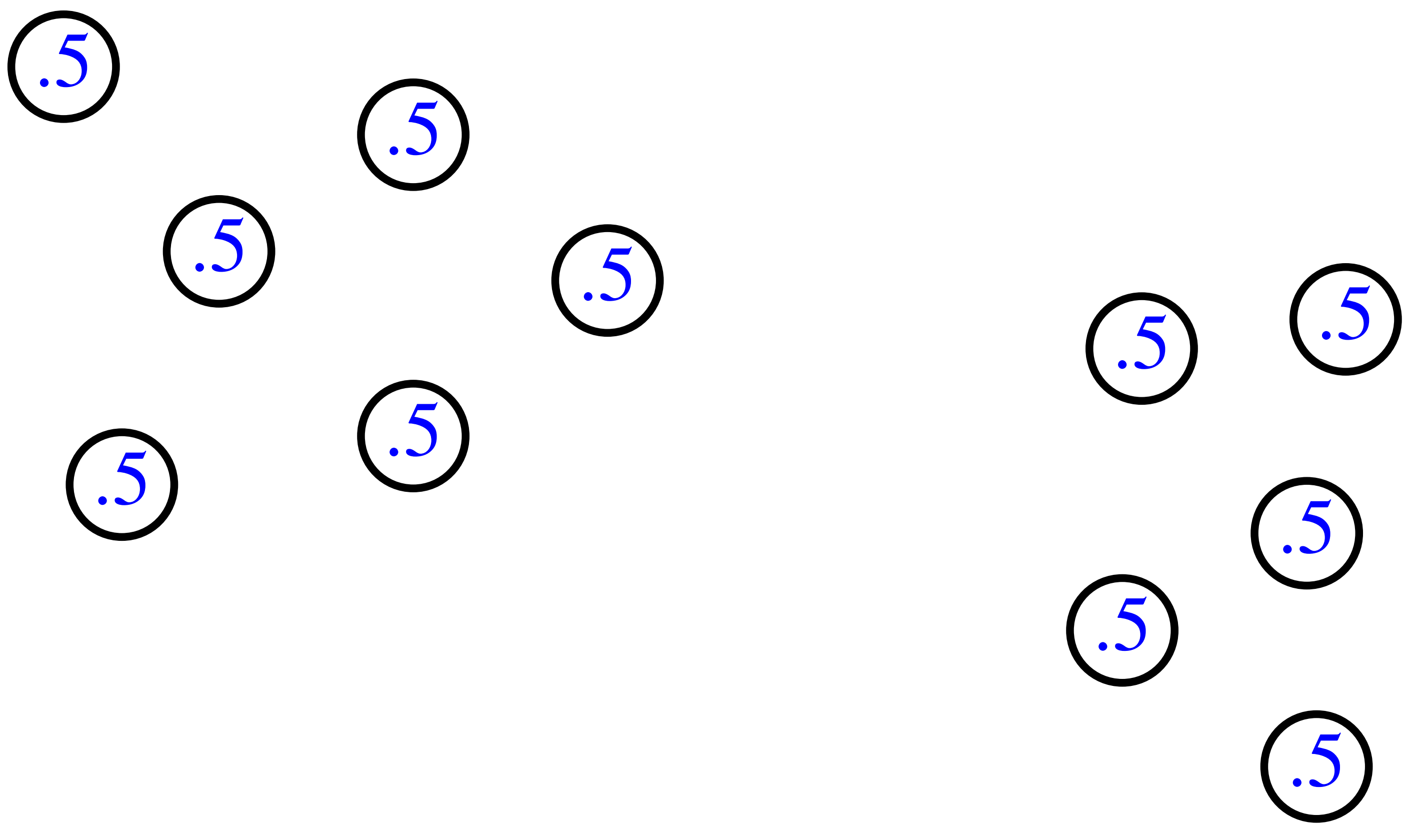
2. Update p_{ij} (the probability that $X_i = j$) by taking the fraction of times that $X_i = j$ for the elite samples

3. Repeat until probabilities all close to 0 or 1

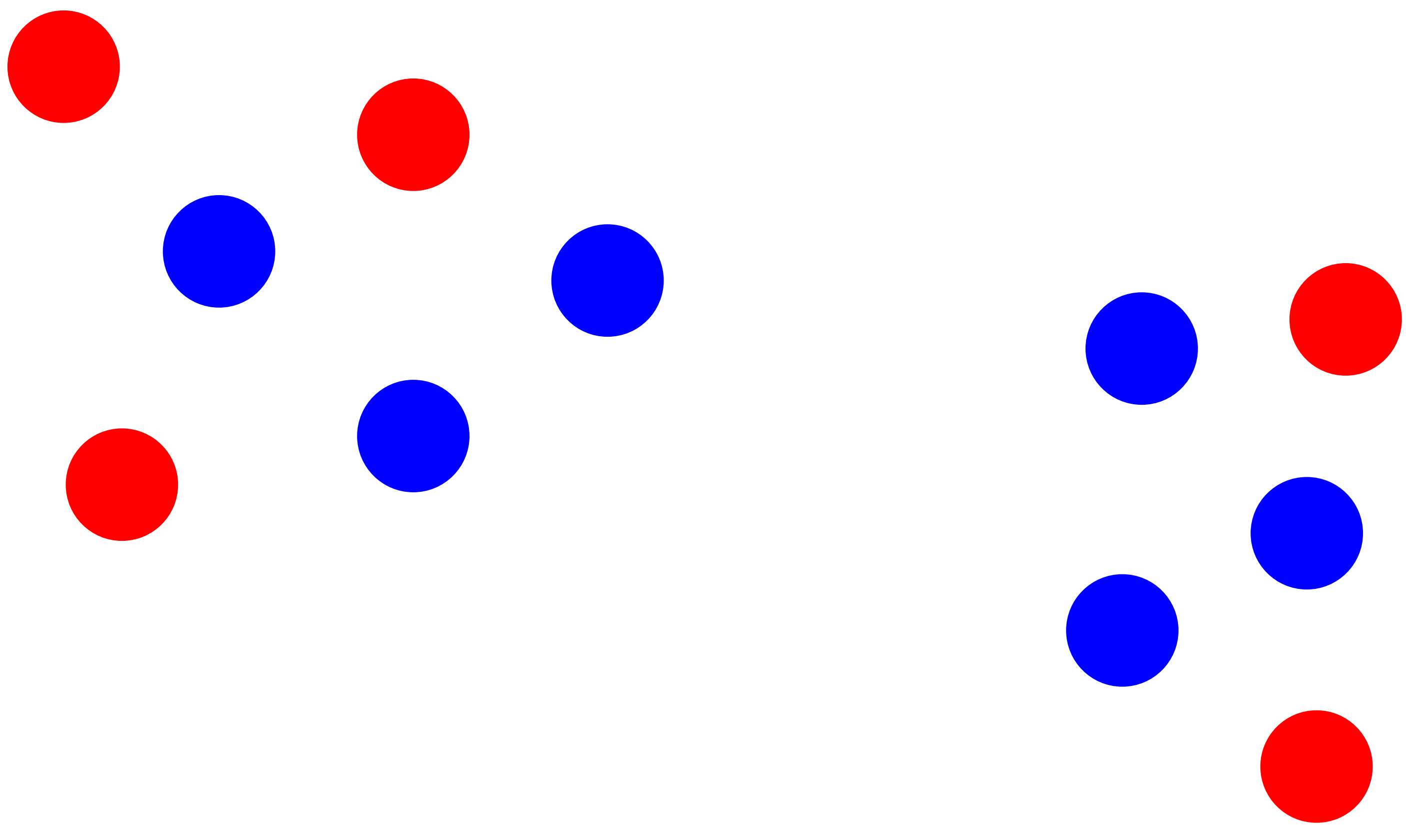
CE Cluster Vector – Example



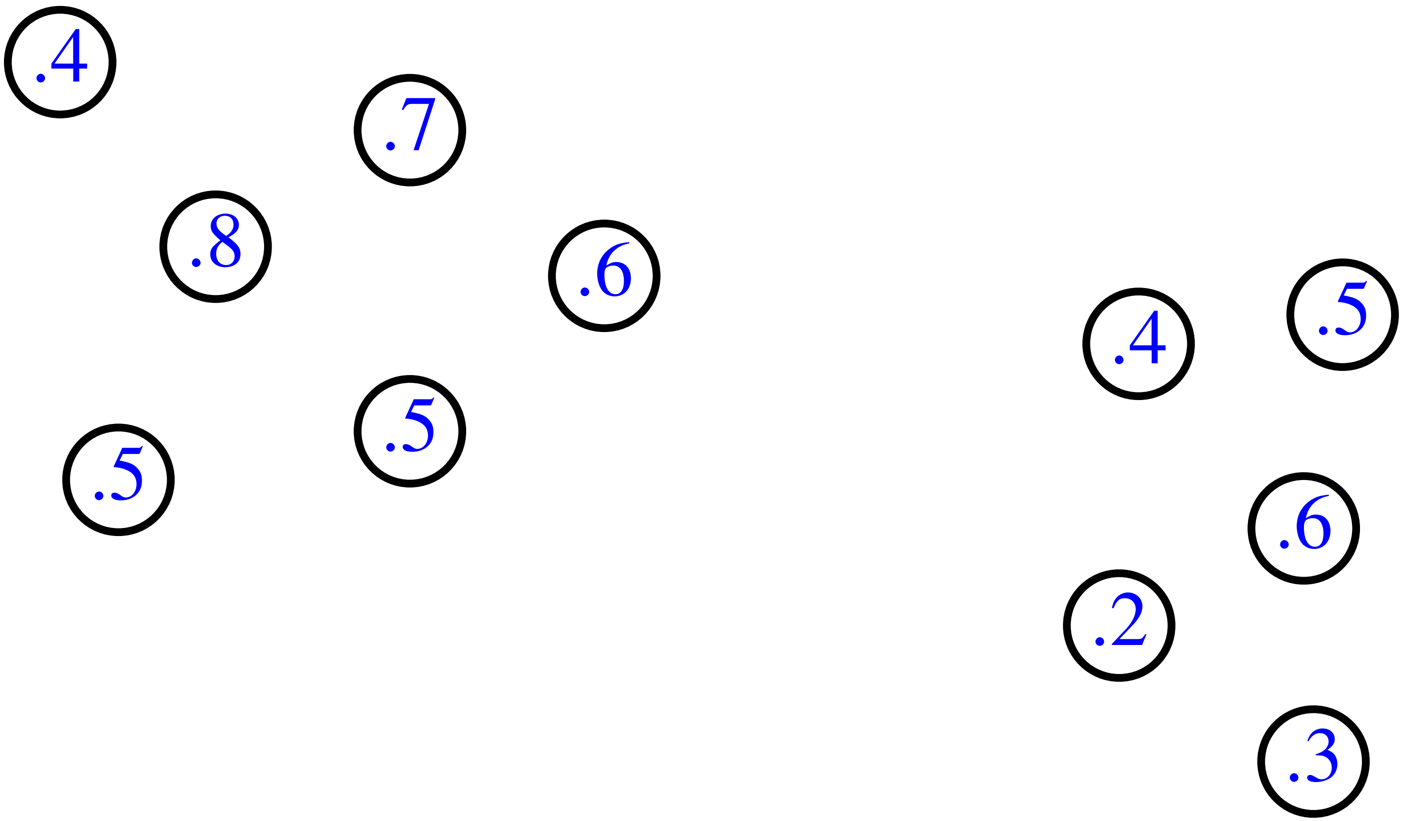
CE Cluster Vector – Example



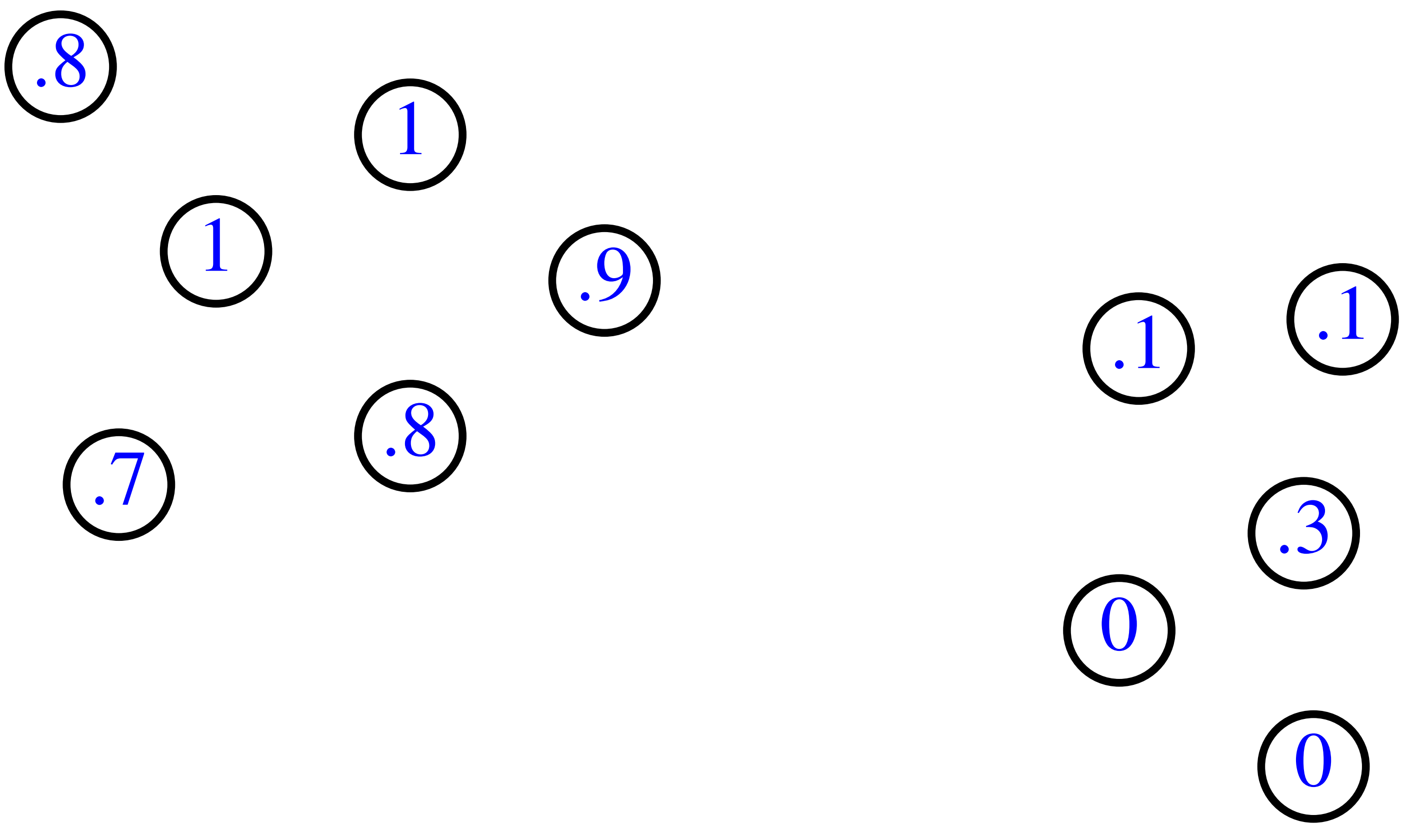
CE Cluster Vector – Example



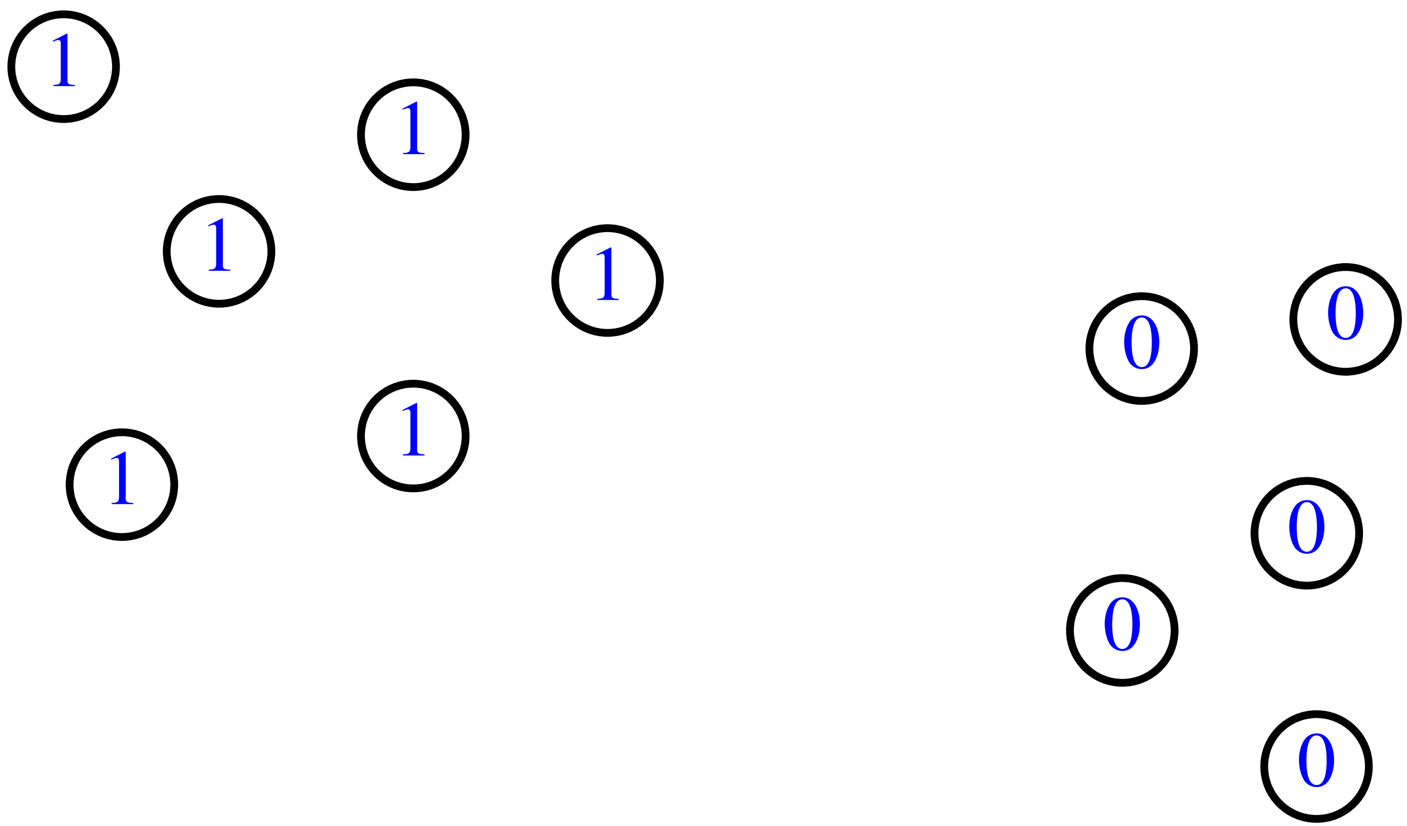
CE Cluster Vector – Example



CE Cluster Vector – Example



CE Cluster Vector – Example



CE with Cluster Centroids

Performance function:

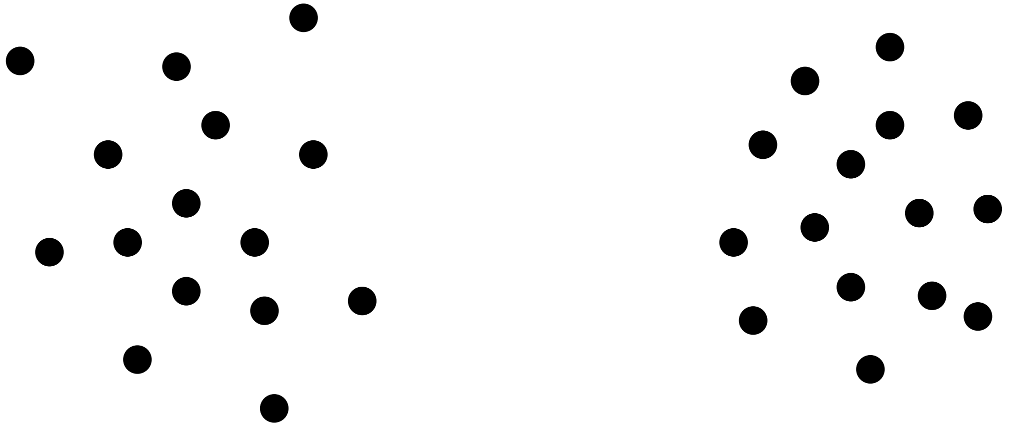
$S(\mathbf{x}) = \sum_{j=1}^{K} \sum_{\mathbf{z} \in R_j} ||\mathbf{z} - \mathbf{c}_j||^2$

1. Generate N sequences of K centroids independently from (typically) Gaussian pdfs

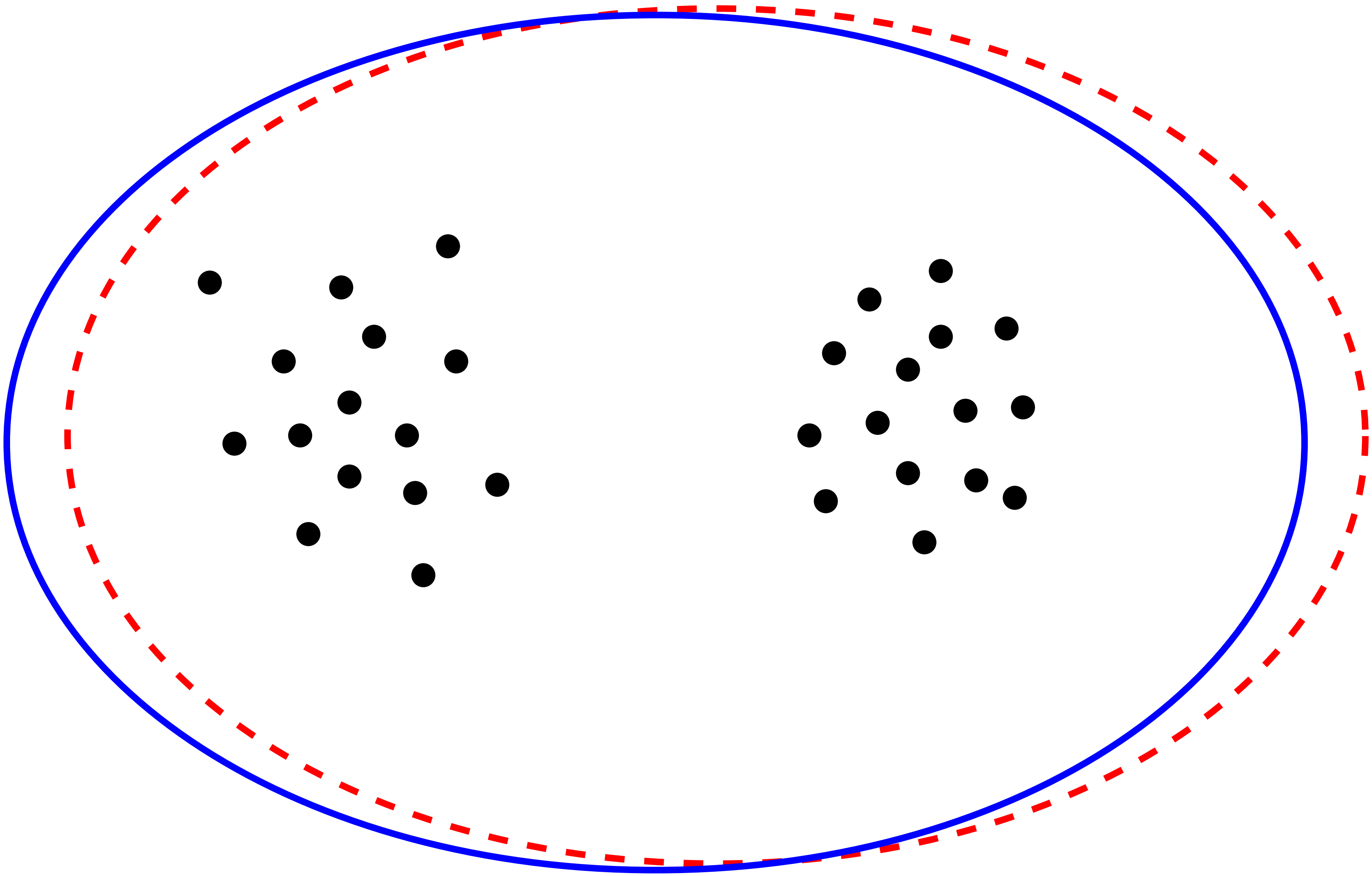
2. Update means and variances of pdfs with corresponding *sample* means and *sample* variances of the elite samples

3. Repeat until variances very small

CE Cluster Centroids – Example

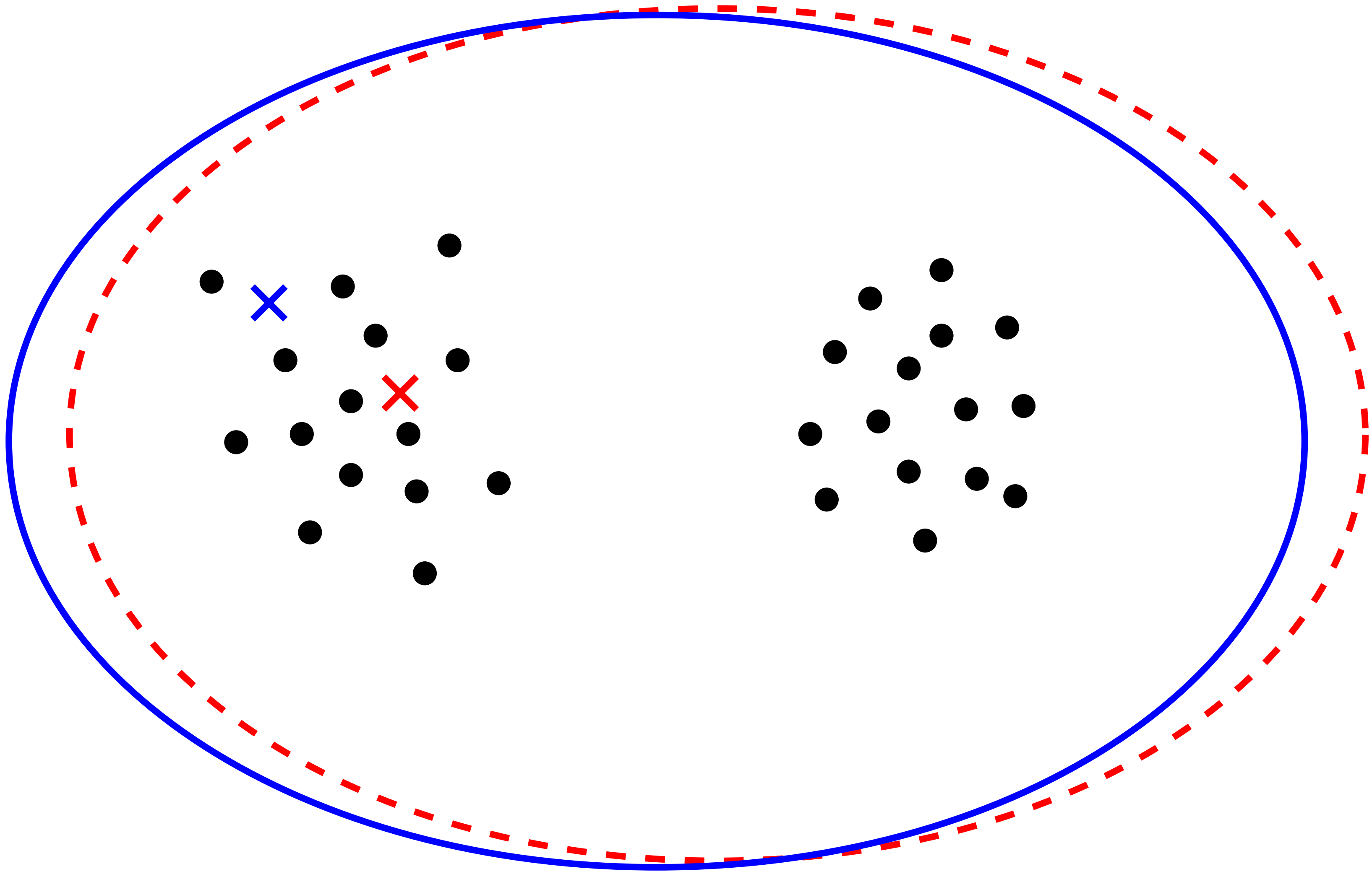


CE Cluster Centroids – Example

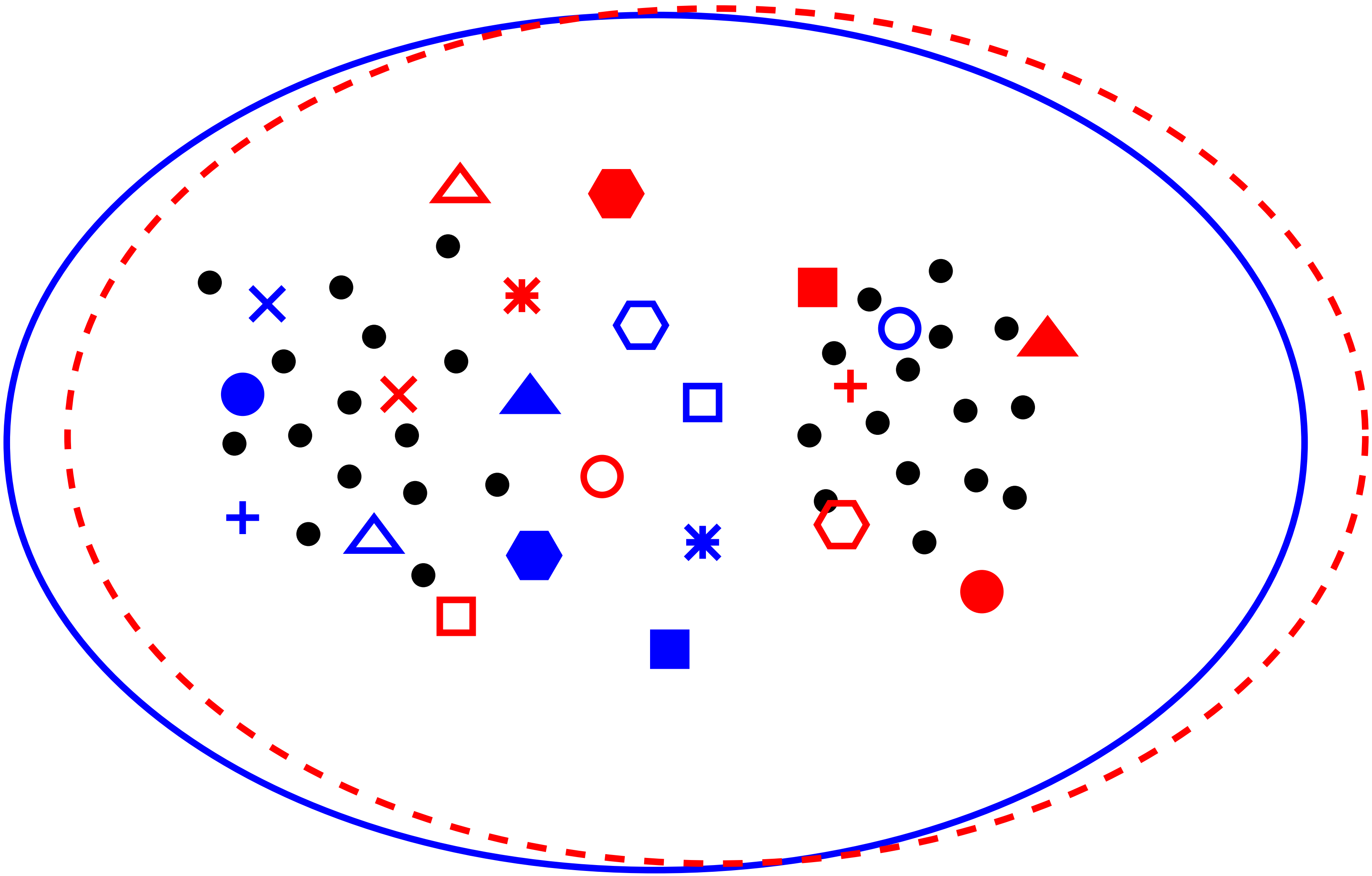


Approach 2

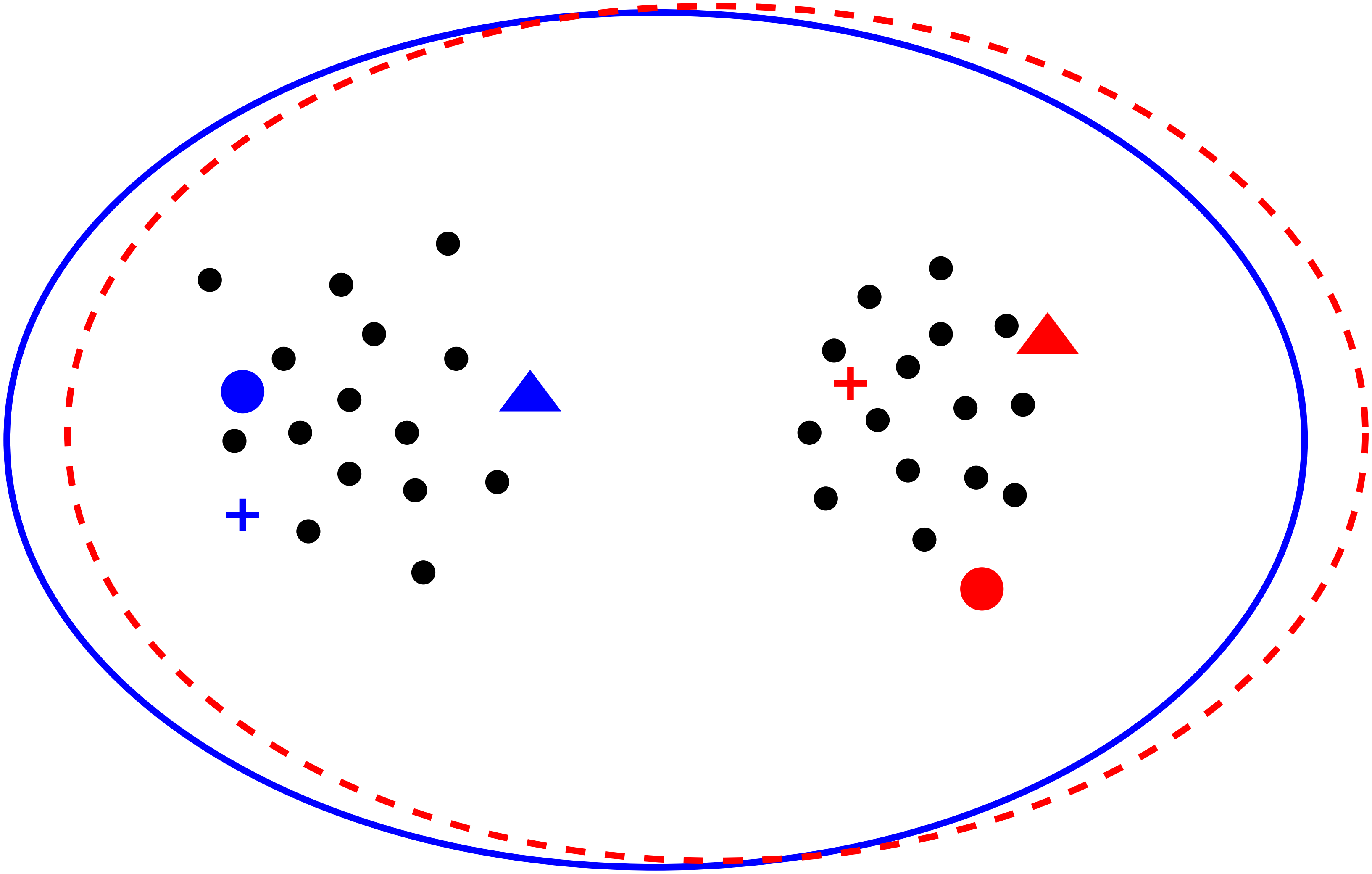
CE Cluster Centroids – Example



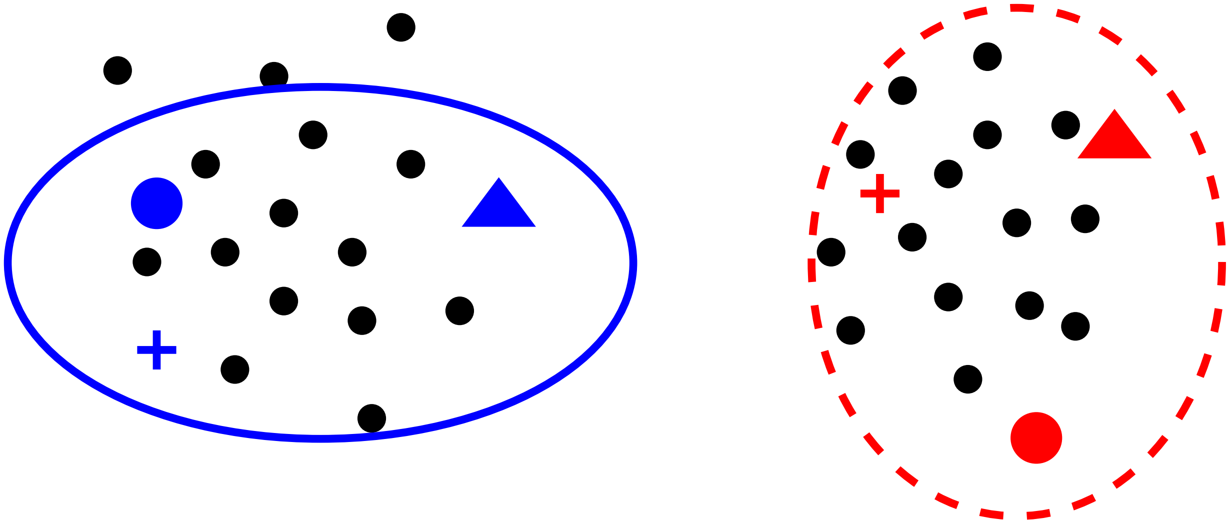
CE Cluster Centroids – Example



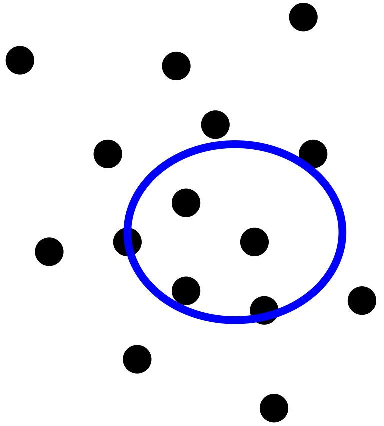
CE Cluster Centroids – Example



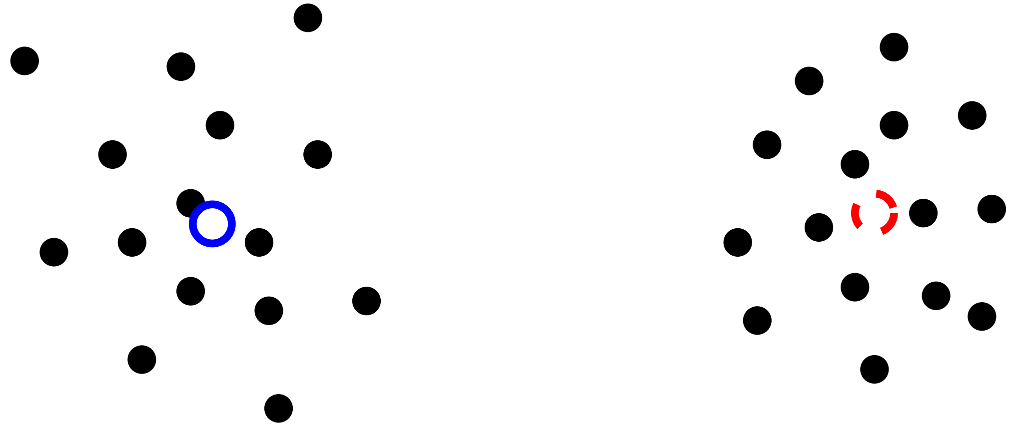
CE Cluster Centroids – Example



CE Cluster Centroids – Example

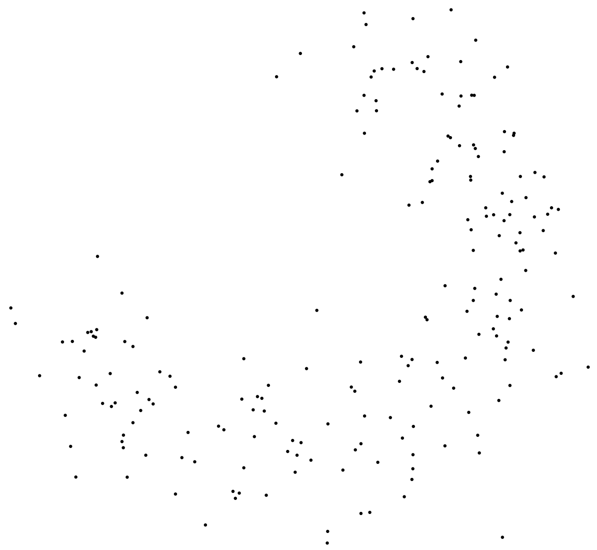


CE Cluster Centroids – Example



Simulations – Sample Results

- Will focus on CE Approach 2 for the rest of this talk
- Setup : CE (Approach 2) vs *K-means* (KM), *Fuzzy K-means* (FKM) and *Linear Vector Quantization* (LVQ).
- Banana data set: 200 Points are scattered around a segment of a circle



Evolution of the Algorithm

Banana Data Set, with $K = 5$ clusters

It.	Worst*	Best	Std. Dev.
1	452.18	377.37	3.00000
2	434.09	395.01	3.20577
3	420.91	366.87	3.38746
4	403.82	356.78	3.16696
5	374.37	336.55	3.30599
6	364.62	333.48	2.94838
7	344.82	325.18	2.53459
8	333.42	313.58	2.22936
9	317.22	302.15	1.58735
10	305.09	295.90	1.15077
11	296.10	292.34	0.65408

It.	Worst*	Best	Std. Dev.
12	291.75	290.31	0.45115
13	289.66	288.55	0.26106
14	288.80	288.50	0.18901
15	288.46	288.27	0.11668
16	288.26	288.18	0.07314
17	288.18	288.14	0.05173
18	288.14	288.13	0.03390
19	288.12	288.12	0.02360
20	288.11	288.11	0.01730
21	288.11	288.11	0.01013
22	288.11	288.11	0.00705

* of elite samples

CE Parameter Setup

- Initial parameter tuple : $(N, N^{\text{elite}}, \alpha) = (800, 20, 0.7)$
- μ_0 : Uniformly drawn from the rectangular area of the data set.
- $\sigma_0 = 14$ (To make initial sampling essentially uniform)
- Stopping : Performance no longer changes within two decimal places or $\max\, \sigma < 10^{-4}$

How Does CE Compare?

Banana Data Set, with K = 5 clusters : each algorithm run 10 times

Approach	T	$\bar{\gamma}_T$	γ^*	$\bar{\varepsilon}$	ε_*	ε^*	CPU
`CE(2)`	49.6	288.49	288.11	0.00	0.00	0.01	26.67
`KM`	9.3	294.31	288.11	0.02	0.01	0.04	0.09
`FKM`	80.6	290.19	288.11	0.01	0.01	0.01	0.14
`LVQ`	17.7	302.81	288.11	0.05	0.01	0.19	0.07

T : Average total number of iterations

$\bar{\gamma}_T$: Averaged solution

γ^* : Best known solution

$\bar{\varepsilon}$: Average relative experimental error w.r.t γ^*

ε^*, ε_* : Largest and smallest relative experimental errors

CPU : Average CPU run time in seconds on a 1.67GHz PC

Summary of Results So Far

CE algorithm is :

- Significantly slower; BUT
- More accurate and consistent than the others
- As number of clusters increases, we see :
  - Efficiency (in terms of $\bar{\varepsilon},\ \varepsilon_*,\ \varepsilon^*$) of CE increases relative to others

Simulations – A Fairer Comparison

We noted :

- CE is slower than the others, so ...
  - Could run the others several times for each run of the CE algorithm
- Next set up :
  - Run CE 10 times, and record the time.
  - Run other algorithms until their time is up.

A Fairer Comparison

Banana Data Set, with $K = 5$ clusters

Approach	Min	Max	Mean	Trials	CPU	Av Its
`CE(2)`	288.11	288.50	288.15	10	211.50	39.3
`KM`	289.29	549.54	302.16	18803	211.59	10.90
`FKM`	290.19	290.19	290.19	5253	211.69	75.70
`LVQ`	289.26	413.96	300.60	7754	211.68	14.96

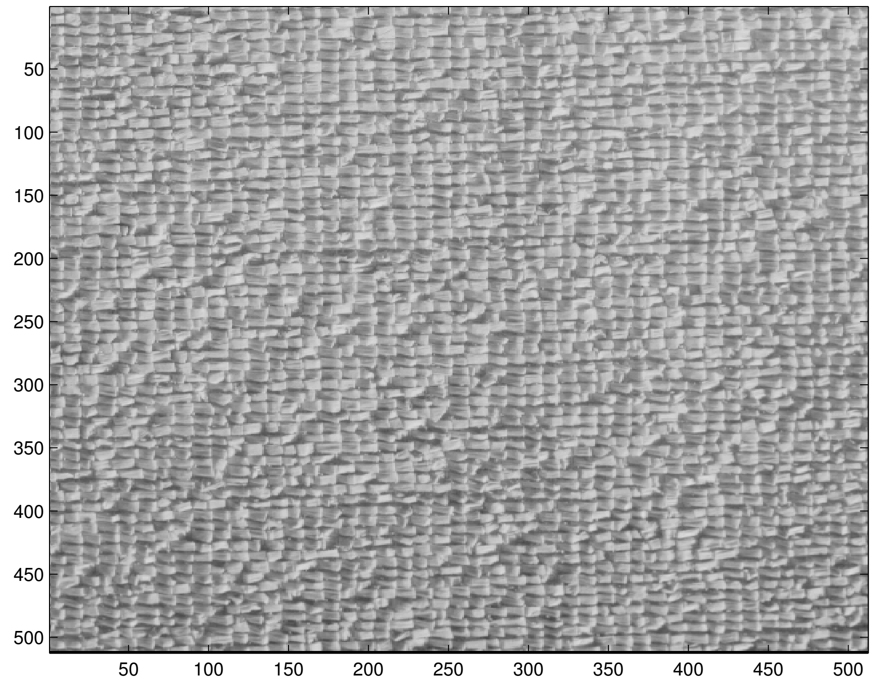
- In this comparison, instructive to look at number of trials and consistency of an approach

Updated Conclusions

CE algorithm is :

- Significantly slower; BUT
- More accurate and consistent than the others; AND
- Outperforms others even over same amount of CPU time

Application – Image Texture Data

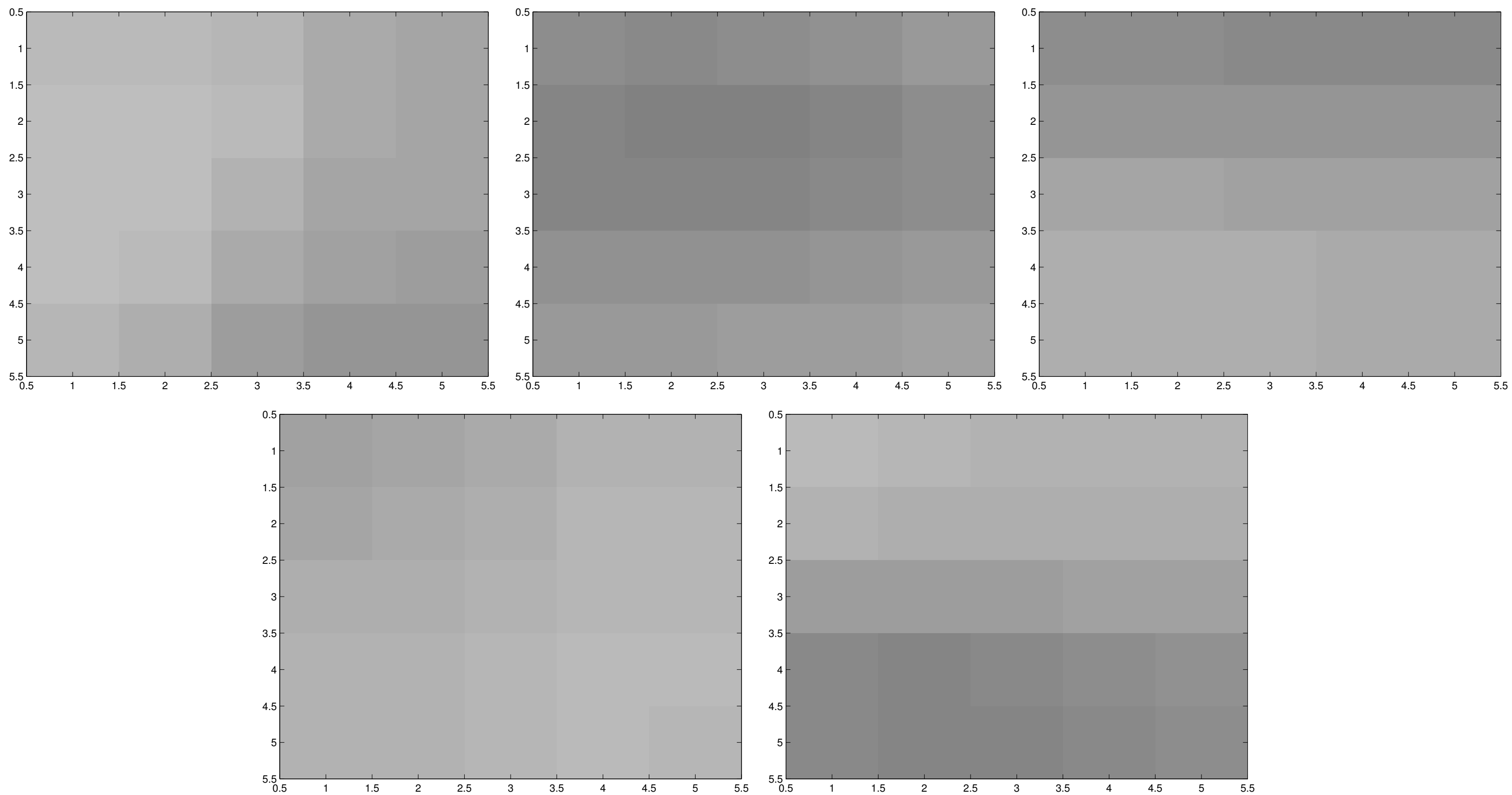


(Raffia texture image, taken from the *USC-SIPI Image Database*)

- Problem: Find five “characteristic” images which “best” distill this test image

Application – Image Texture Data

The best five characteristic images found by CE.



Application – Image Texture Data

Raffia Image Data Set, with $K = 5$ clusters

Approach	Min	Mean	Max	$\bar{\varepsilon}$	ε_*	ε^*	Trials	CPU	Av Its
CE(2)	83.67	84.21	85.45	0.0065	0	0.0213	10	4057.3	19.2
KM	83.81	84.61	95.80	0.0112	0.0017	0.1450	70363	4057.6	11.67
FKM	91.93	91.93	91.93	0.0987	0.0987	0.0987	13806	4058.7	174.03
LVQ	83.78	84.76	95.81	0.0130	0.0013	0.1451	61847	4057.7	10.12

Summing Up

- We presented application of the CE method to clustering and vector quantization problems by:
  - Generation of random clusters using either
    - *Approach 1* : Independent k-point distributions or,
    - *Approach 2* : Independent Gaussian distributions
  - Followed by updating the associated parameters using cross-entropy minimization.
- Simulations suggest
  - CE Algorithm reliable
  - May be considered as alternative to the standard clustering methods

Directions for Future Research

- Establishing convergence of CE Algorithm for finite sampling
- Establishing confidence regions for the optimal solution
- Application of parallel optimization techniques to the proposed methodology

Acknowledgments

Acknowledgment: We are most grateful to Uri Dubin for his contributions

This work was partially supported by

AUSTRALIAN RESEARCH COUNCIL
Centre of Excellence for Mathematics
and Statistics of Complex Systems