



Kernel Clustering

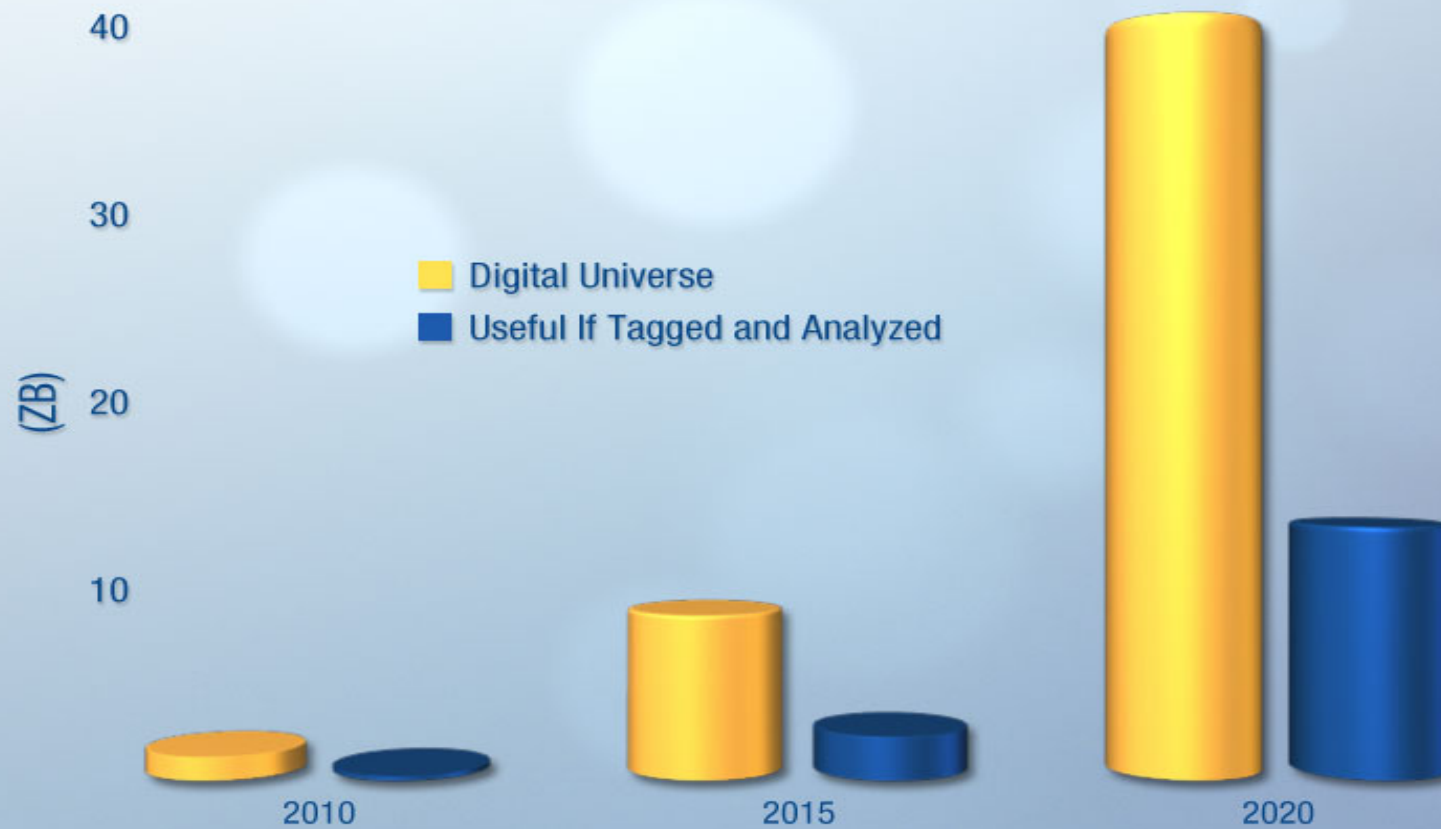
CSE 902
Radha Chitta

Outline

- Data analysis
- Clustering
- Kernel Clustering
 - Kernel K-means and Spectral Clustering
- Challenges and Solutions

Data Analysis

Opportunity for Big Data



Learning Problems



Learning Problems

➤ Supervised Learning (Classification)



This is a cat.



This is a dog.



What is this?

➤ Semi-supervised learning



This is a cat.



This is a dog.



Learning Problems

- **Clustering** into “known” number of clusters



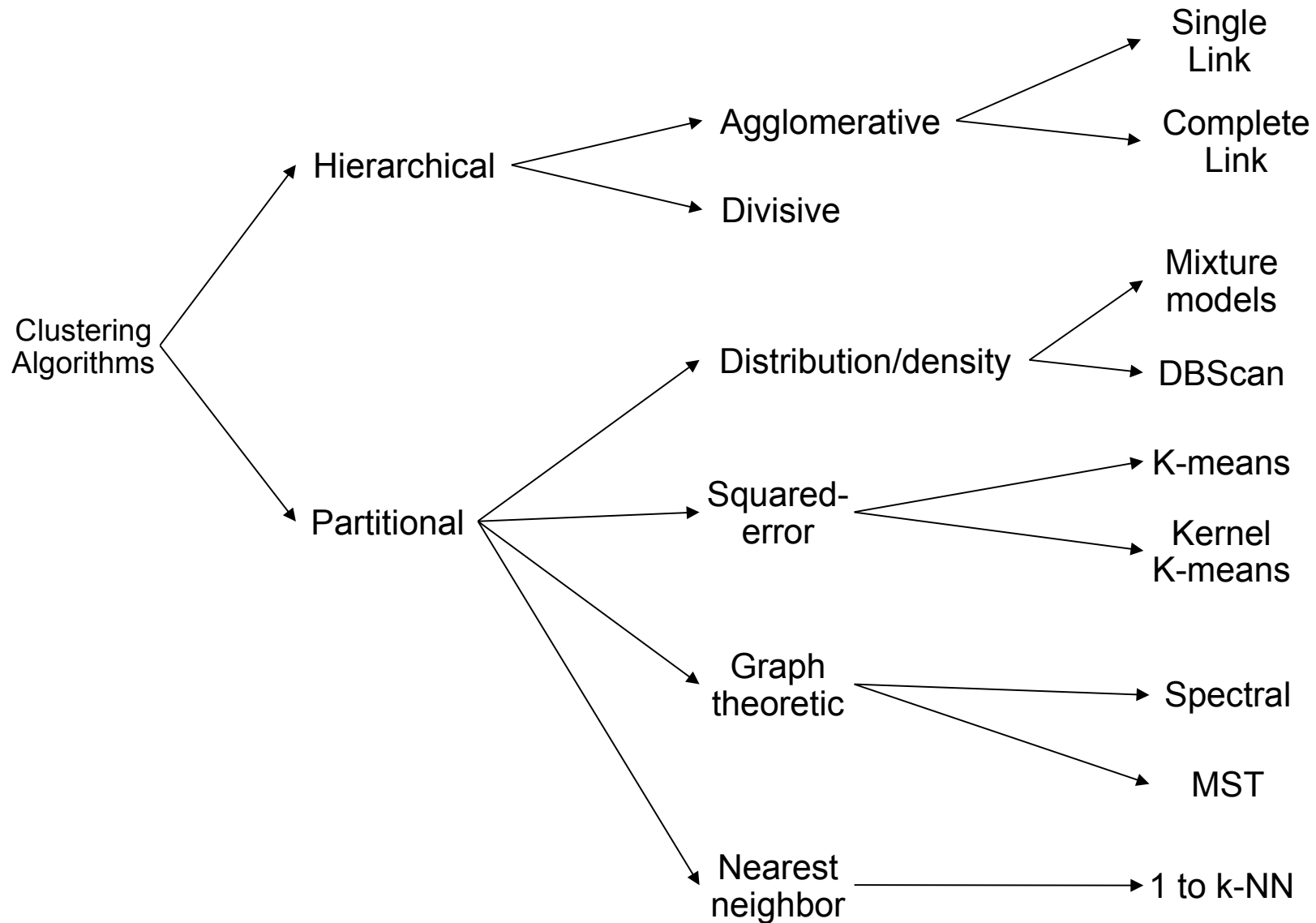
There are two objects in this set of images. Separate them into “two” groups.

- **Completely unsupervised learning**

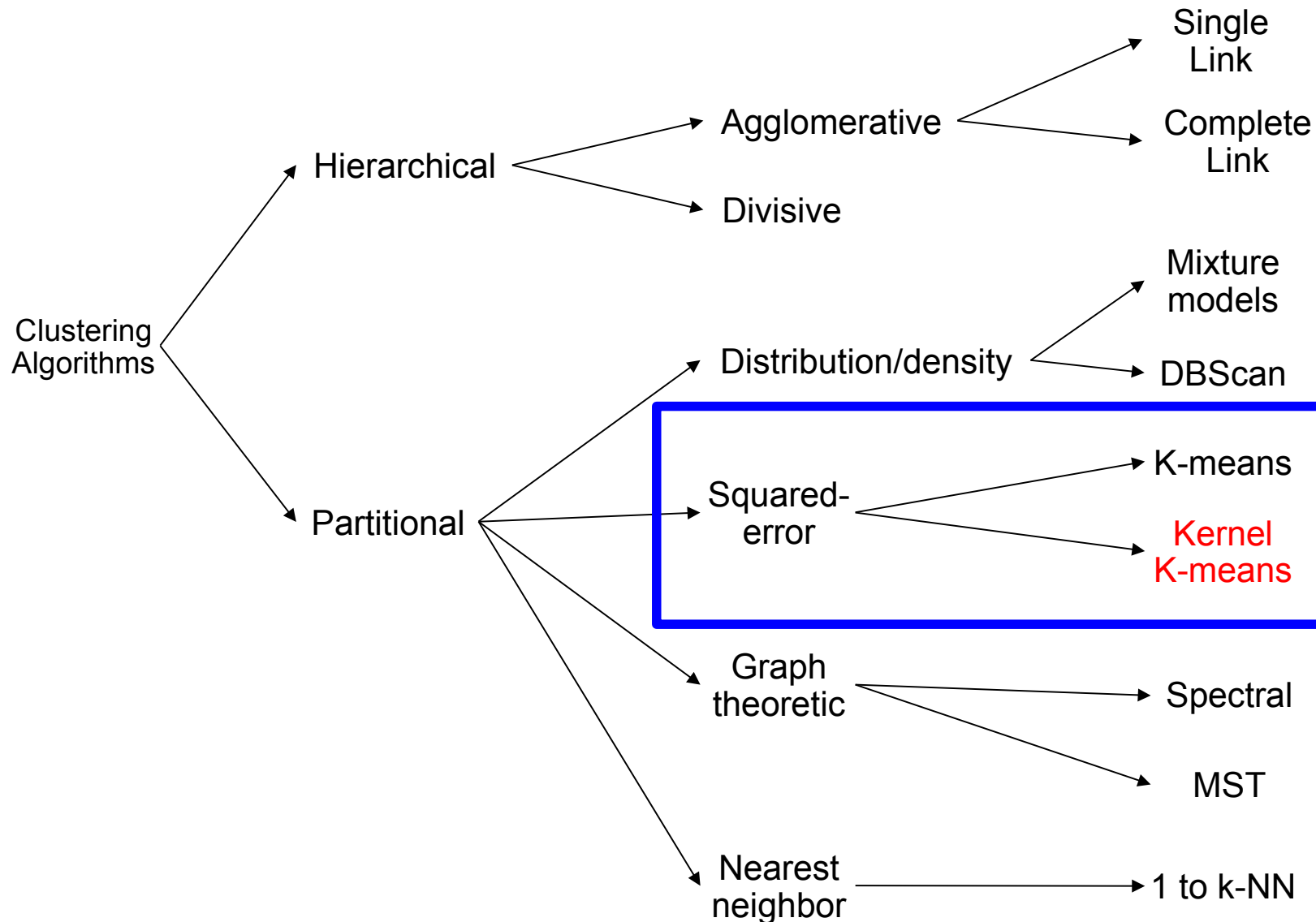


No additional information. Just images

Clustering Algorithms



Clustering Algorithms



The K-means problem

Given data set

$$D = \{x_1, x_2, \dots, x_n\}$$

find **C** representative points in the data set to minimize
sum of squared error

$$\min_U \sum_{k=1}^C \sum_{i=1}^n U_{ki} \|c_k - x_i\|_2^2$$

← Euclidean distance

1 if x_i belongs to
cluster k ; 0 otherwise

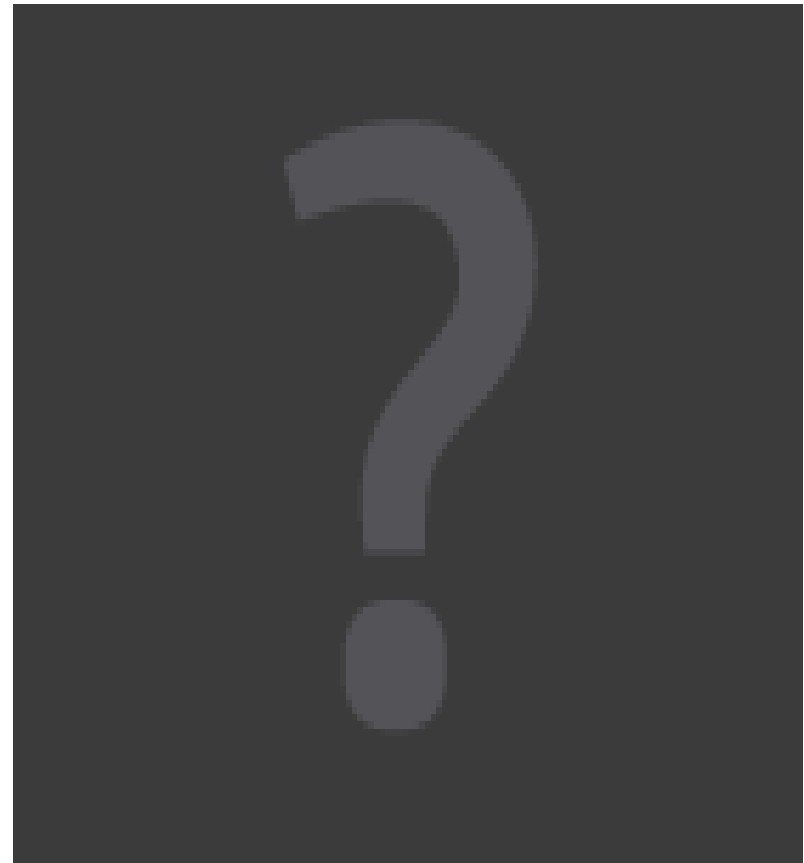
Cluster centers

0-1 integer programming problem: **NP-Complete**

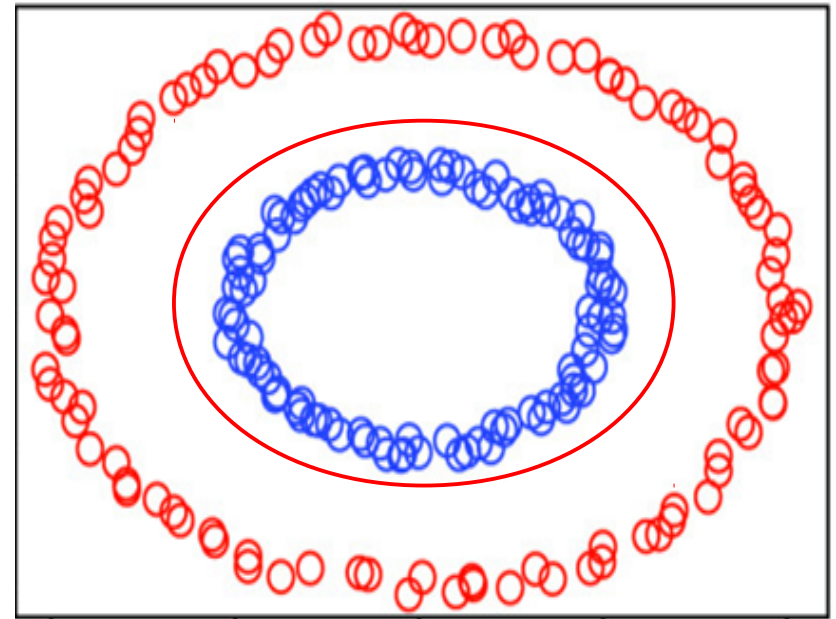
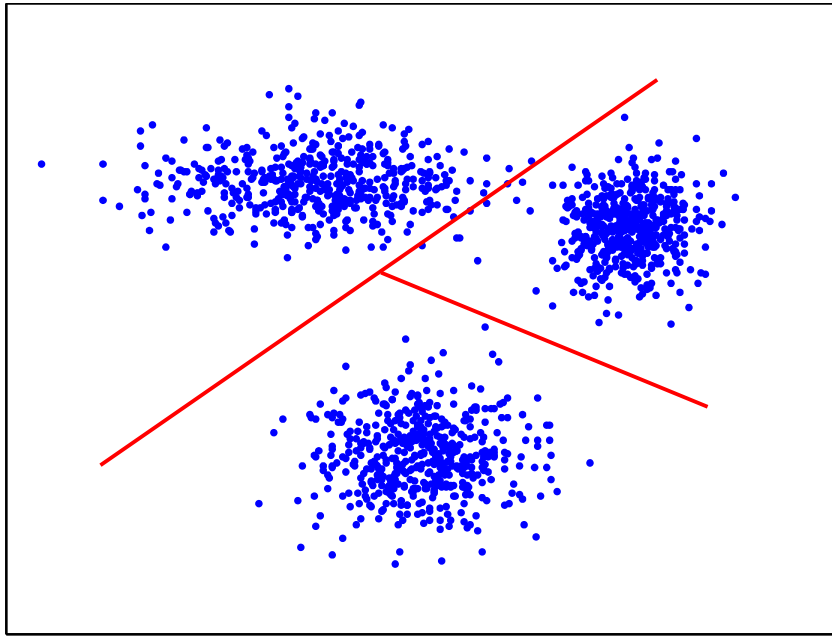
Lloyd's K-means algorithm

- Initialize labels.
- Repeat until convergence
 - Find centers of clusters
 - Assign points to the closest center.

$O(nCd)$ running time complexity



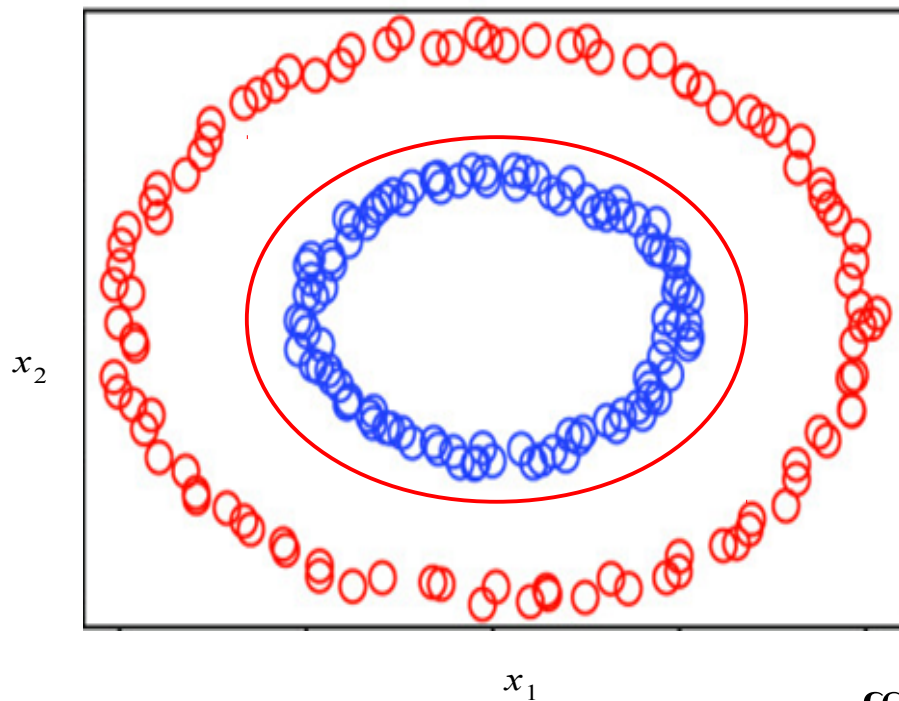
K-means doesn't always work!



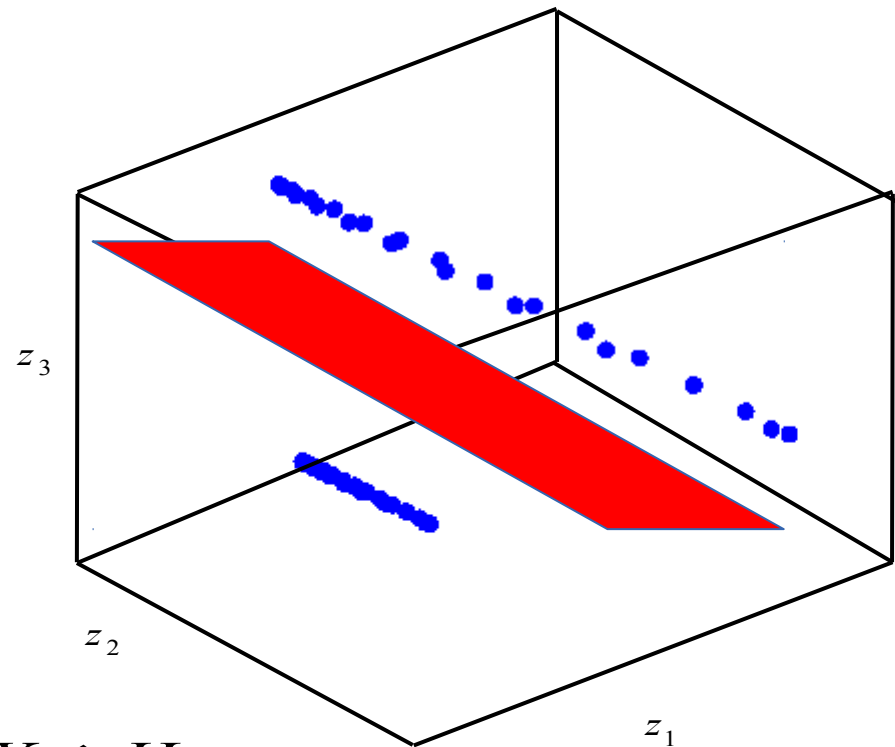
Euclidean distance assumes unit covariance and linear separability

Non-linear separation

Any data set is linearly separable in sufficiently high dimensional space



$$\varphi : X \rightarrow H$$



Polynomial

$$\varphi(x) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$

Dot product kernels

Euclidean distance in H

$$\|\varphi(\mathbf{x}) - \varphi(\mathbf{y})\|_2^2 = \underbrace{\varphi(\mathbf{x})^T \varphi(\mathbf{x})}_{\|\varphi(\mathbf{x})\|_2^2} - 2 \underbrace{\varphi(\mathbf{x})^T \varphi(\mathbf{y})}_{\langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle} + \underbrace{\varphi(\mathbf{y})^T \varphi(\mathbf{y})}_{\|\varphi(\mathbf{y})\|_2^2}$$

Polynomial map

$$\varphi([x_1, x_2]) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$

$$\varphi(\mathbf{x})^T \varphi(\mathbf{y}) = \begin{pmatrix} x_1^2 & \sqrt{2} x_1 x_2 & x_2^2 \end{pmatrix} \begin{pmatrix} y_1^2 \\ \sqrt{2} y_1 y_2 \\ y_2^2 \end{pmatrix} = (x^T y)^2$$

Dot product kernels

Euclidean distance in H

$$\|\varphi(\mathbf{x}) - \varphi(\mathbf{y})\|_2^2 = \underbrace{\varphi(\mathbf{x})^T \varphi(\mathbf{x})}_{\|\varphi(\mathbf{x})\|_2^2} - 2 \underbrace{\varphi(\mathbf{x})^T \varphi(\mathbf{y})}_{\langle \varphi(\mathbf{x}), \varphi(\mathbf{y}) \rangle} + \underbrace{\varphi(\mathbf{y})^T \varphi(\mathbf{y})}_{\|\varphi(\mathbf{y})\|_2^2}$$

Polynomial map

$$\varphi([x_1, x_2]) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$

$$\varphi(\mathbf{x})^T \varphi(\mathbf{y}) = \begin{pmatrix} x_1^2 & \sqrt{2} x_1 x_2 & x_2^2 \end{pmatrix} \begin{pmatrix} y_1^2 \\ \sqrt{2} y_1 y_2 \\ y_2^2 \end{pmatrix} = (x^T y)^2$$

Dot product kernels

Euclidean distance in H

$$\|\varphi(\mathbf{x}) - \varphi(\mathbf{y})\|_2^2 = \underbrace{\varphi(\mathbf{x})^T \varphi(\mathbf{x})}_{\|\varphi(\mathbf{x})\|_2^2} - 2 \underbrace{\varphi(\mathbf{x})^T \varphi(\mathbf{y})}_{\kappa(\mathbf{x}, \mathbf{y})} + \underbrace{\varphi(\mathbf{y})^T \varphi(\mathbf{y})}_{\|\varphi(\mathbf{y})\|_2^2}$$

Polynomial map

$$\varphi(\mathbf{x}) = (x_1^2, \sqrt{2} x_1 x_2, x_2^2)$$

$$\varphi(\mathbf{x})^T \varphi(\mathbf{y}) = \begin{pmatrix} x_1^2 & \sqrt{2} x_1 x_2 & x_2^2 \end{pmatrix} \begin{pmatrix} y_1^2 \\ \sqrt{2} y_1 y_2 \\ y_2^2 \end{pmatrix} = \boxed{(x^T y)^2}$$

Kernel function

$$\kappa(\mathbf{x}, \mathbf{y})$$

Mercer kernels

There exists a mapping $\varphi(x)$ and an expansion

$$\kappa(x, y) = \varphi(x)^T \varphi(y)$$

iff for any $g(x)$ such that

$$\int g(x)^2 dx$$

is finite then

$$\int \kappa(x, y) g(x) g(y) dx dy \geq 0$$

$$d_H^2(x, y) = \kappa(x, x) - 2\kappa(x, y) + \kappa(y, y)$$

Kernels

- Polynomial (Linear)

$$\kappa(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y})^p$$

- Gaussian

$$\kappa(\mathbf{x}, \mathbf{y}) = \exp(-\lambda \|\mathbf{x} - \mathbf{y}\|_2^2)$$

- Chi-square

$$\kappa(\mathbf{x}, \mathbf{y}) = 1 - \sum_{i=1}^d \frac{(x_i - y_i)^2}{0.5(x_i + y_i)}$$

- Histogram intersection

$$\kappa(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^m \min(\text{hist}(\mathbf{x}), \text{hist}(\mathbf{y}))$$

Kernel K-means

Finding C clusters in Hilbert (feature) space H to minimize sum of squared error

$$\min_U \sum_{k=1}^C \sum_{i=1}^n U_{ki} \|c_k - \varphi(x_i)\|_H^2$$

$$\min_U \sum_{i=1}^n \sum_{k=1}^C U_{ki} \left[\kappa(x_i, x_i) - \frac{2}{n_k} \sum_{j=1}^n U_{kj} \kappa(x_i, x_j) + \frac{1}{n_k^2} \sum_{j=1}^n \sum_{l=1}^n U_{kj} U_{kl} \kappa(x_j, x_l) \right]$$

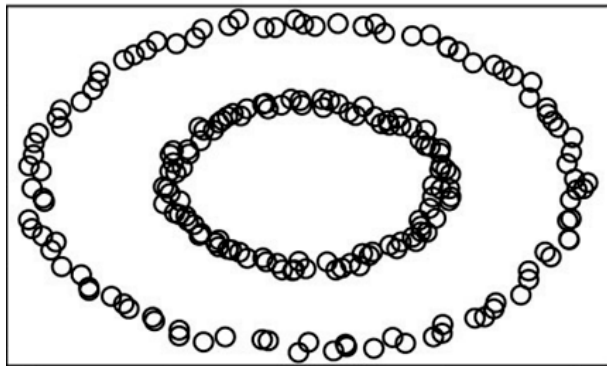
Calculate kernel (Gram) matrix and run K-means in H

$$K = [\kappa(x_i, x_j)]_{n \times n}; x_i, x_j \in D$$

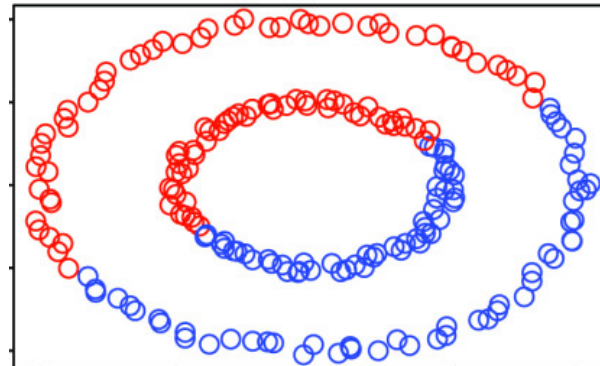
$$\max \text{trace}(UKU')$$

K-means Vs Kernel K-means

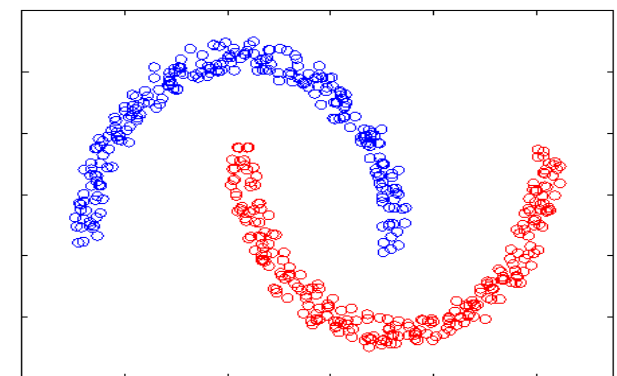
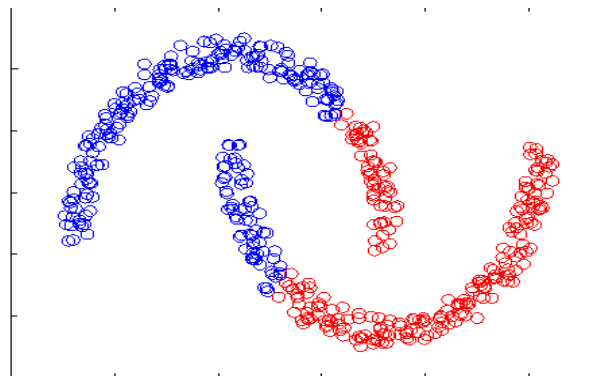
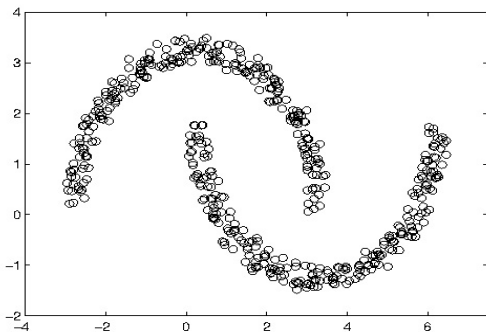
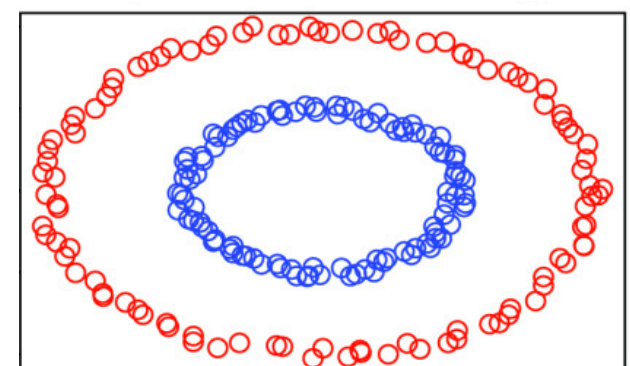
Data



K-means



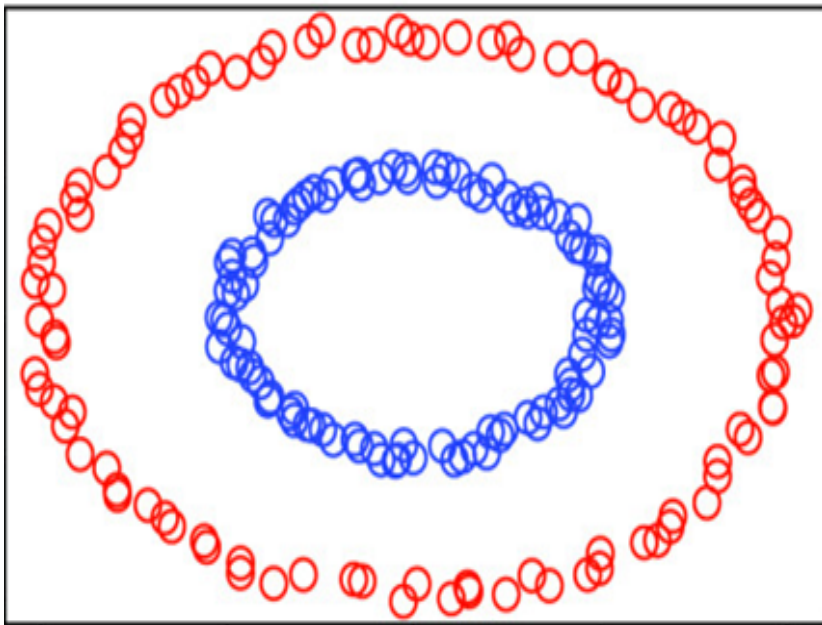
Kernel K-means



Kernel K-means is able to find “complex” clusters.

Spectral Clustering

Represents the data using the top K eigenvectors and obtain the clusters



Equivalent to Weighted Kernel K-means

$$\max \text{trace} \left(U D^{-1/2} K D^{-1/2} U' \right)$$

Kernel K-means: Challenges

- **Scalability**
 - $O(n^2)$ complexity to calculate kernel
 - More expensive than K-means
- **Out-of-sample clustering**
 - No explicit representation for the centers.
 - Expensive to assign a new point to a cluster.
- **Kernel selection**
 - The best kernel is application and data-dependent.
 - Wrong kernel can lead to results worse than K-means

Scalability

No. of Objects (n)	No. of operations	
	K-means	Kernel K-means
	$O(nCd)$	$O(n^2C)$
1M	10^{11} (3.2)	10^{15}
10M	10^{12} (34.9)	10^{17}
100M	10^{13} (5508.5)	10^{19}
1B	10^{14}	10^{21}

$d = 100; C=10$

No. of Objects (n)	No. of operations	
	K-means	Kernel K-means
	$O(nCd)$	$O(n^2C)$
1M	10^{13} (6412.9)	10^{16}
10M	10^{14}	10^{18}
100M	10^{15}	10^{20}
1B	10^{16}	10^{22}

$d = 10,000; C=10$

* Runtime in seconds on Intel Xeon 2.8 GHz processor using 12 GB memory

A petascale supercomputer (IBM Sequoia, June 2012) with ~1 exabyte memory is needed to kernel cluster billions of data points!

Scalability

- Reordered clustering
 - Reordered kernel K-means
- Distributed clustering
 - Parallel kernel K-means
- Sampling based approximations
 - Kernel matrix approximation
 - Non-linear Random Projections

Reordered kernel K-means

A Large scale clustering scheme for kernel K-means, Zhang and Rudnicky, ICPR 2002

- Distance

$$d^2(x_i, c_k) = \kappa(x_i, x_i) - \frac{2}{n_k} \sum_{j=1}^n U_{kj} \kappa(x_i, x_j) + \frac{1}{n_k^2} \sum_{j=1}^n \sum_{l=1}^n U_{kj} U_{kl} \kappa(x_j, x_l)$$

- Reorder the clustering process such that only a small portion of the kernel matrix is required at a time.
- Store the full kernel matrix on disk and load part of it into the memory.

Reordered kernel K-means

A Large scale clustering scheme for kernel K-means, Zhang and Rudnicky, ICPR 2002

- Update incrementally using the kernel portion available in memory.

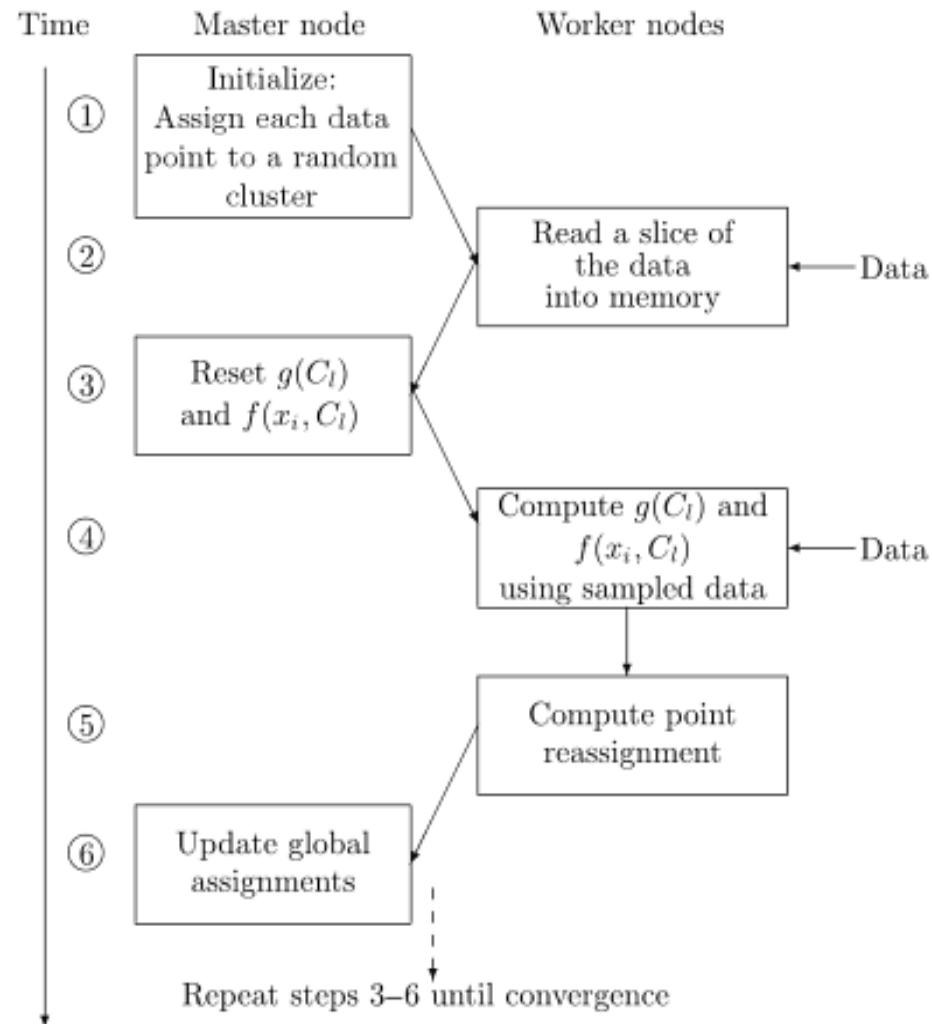
$$f(x_i, c_k) = \frac{-2}{n_k} \sum_{j=1}^n U_{kj} \mathbf{K}(x_i, x_j)$$

$$g(c_k) = \frac{1}{n_k} \sum_{j=1}^n \sum_{l=1}^n U_{kj} U_{kl} \mathbf{K}(x_j, x_l)$$

- Combine values to obtain final partition.

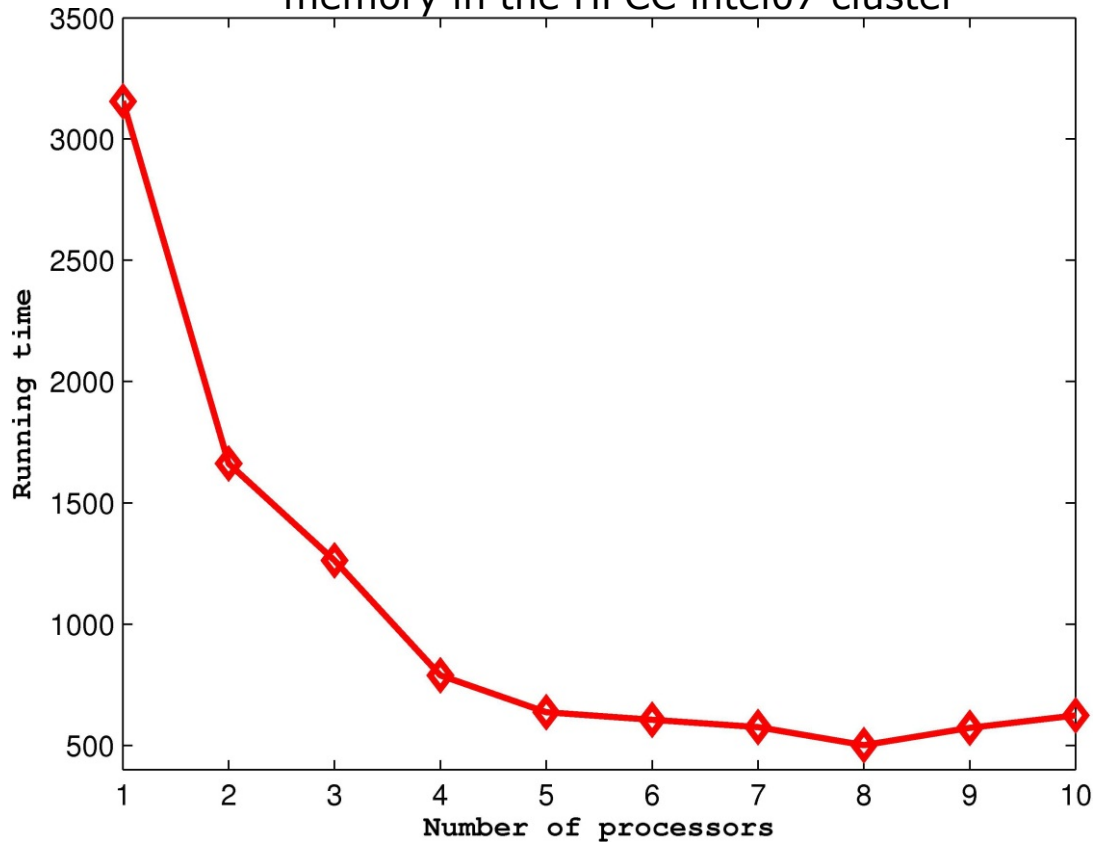
Parallel Kernel K-means

Scaling Up Machine Learning: Parallel and Distributed Approaches, Bekkerman et. al. 2012



Parallel Kernel K-means

K-means on 1 billion 2-D points with 2 clusters
2.3 GHz quad-core Intel Xeon processors, with 8GB
memory in the HPCC intel07 cluster



Number of processors	Speedup
2	1.9
3	2.5
4	3.8
5	4.9
6	5.2
7	5.5
8	6.3
9	5.5
10	5.1

Network communication cost increases with no. of processors

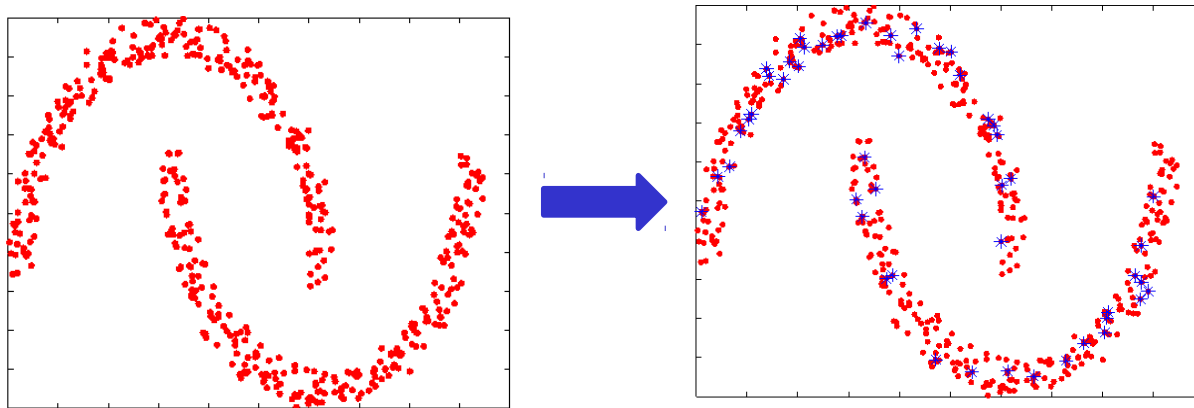
Approx. Kernel K-means

Approximate Kernel k-means: Solution to Large Scale Kernel Clustering, KDD 2011

- Reordering does not reduce runtime complexity, only memory; parallelization does not help reduce runtime and memory complexity
- **Kernel approximation** Use sampling to avoid calculating the full kernel matrix
- Express the cluster centers in terms of sampled points

Approx. Kernel K-means

Approximate Kernel k-means: Solution to Large Scale Kernel Clustering, KDD 2011



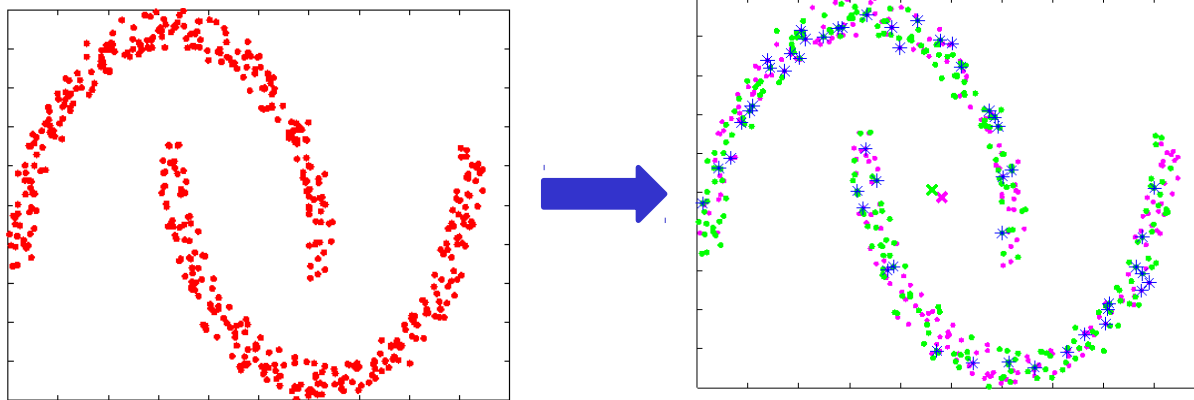
Randomly sample m points $\{y_1, y_2, \dots, y_m\}$, $m \ll n$ and compute the kernel similarity matrices

K_A ($m \times m$) and K_B ($n \times m$)

$$(K_A)_{ij} = \varphi(y_i)^T \varphi(y_j) \quad (K_B)_{ij} = \varphi(x_i)^T \varphi(y_j)$$

Approx. Kernel K-means

Approximate Kernel k-means: Solution to Large Scale Kernel Clustering, KDD 2011



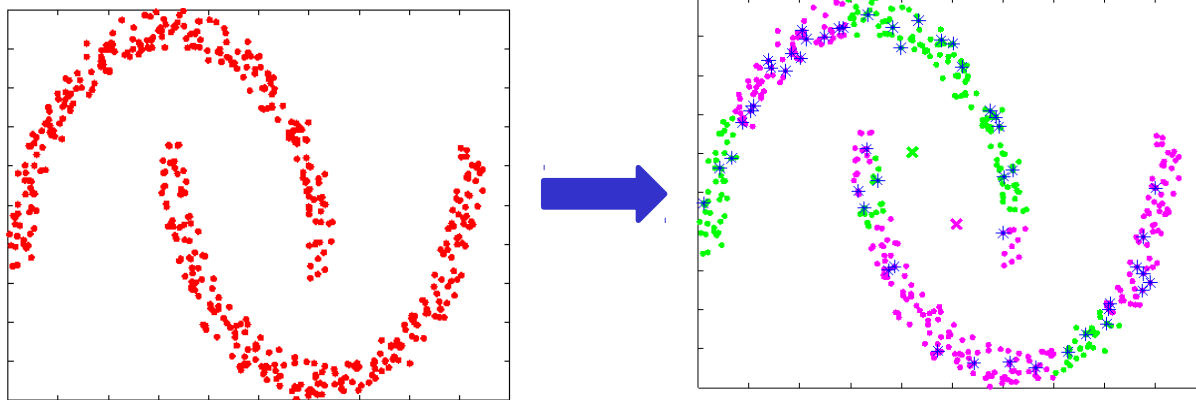
Iteratively optimize for the cluster centers

$$\min_U \max_{\alpha} \sum_{k=1}^C \sum_{i=1}^n U_{ik} \left\| \varphi(x_i) - \sum_{j=1}^m \alpha_{jk} \varphi(y_j) \right\|$$

c_k

Approx. Kernel K-means

Approximate Kernel k-means: Solution to Large Scale Kernel Clustering, KDD 2011



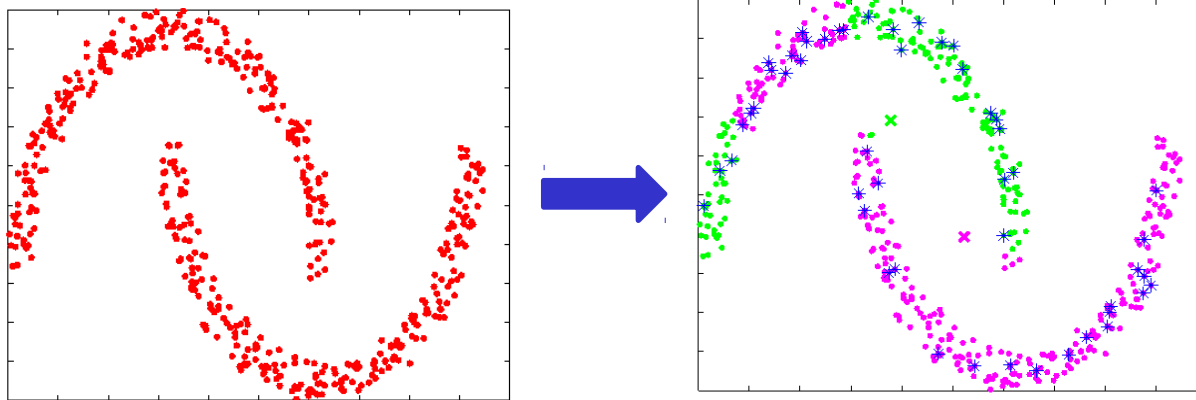
Iteratively optimize for the cluster centers

$$\min_U \max_{\alpha} \sum_{k=1}^C \sum_{i=1}^n U_{ik} \left\| \varphi(x_i) - \sum_{j=1}^m \alpha_{jk} \varphi(y_j) \right\|$$

c_k

Approx. Kernel K-means

Approximate Kernel k-means: Solution to Large Scale Kernel Clustering, KDD 2011



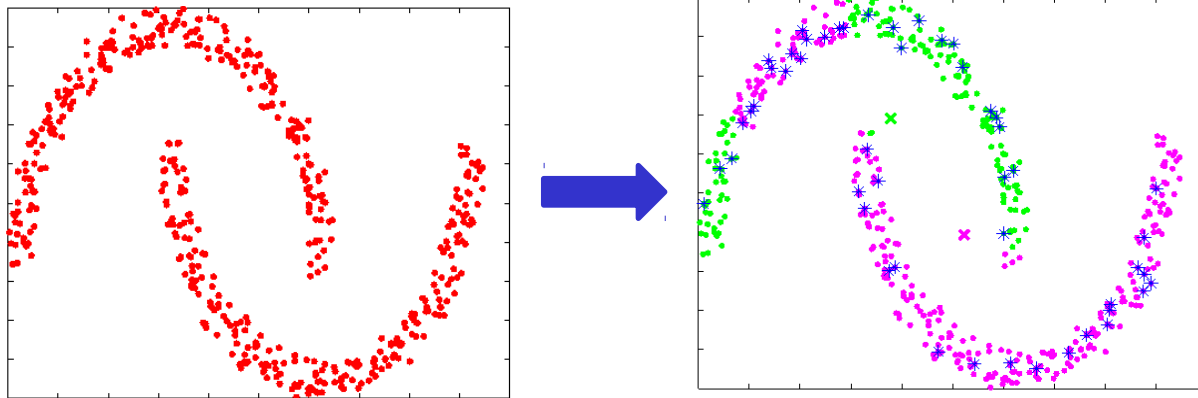
Iteratively optimize for the cluster centers

$$\min_U \max_{\alpha} \sum_{k=1}^C \sum_{i=1}^n U_{ik} \left\| \varphi(x_i) - \sum_{j=1}^m \alpha_{jk} \varphi(y_j) \right\|$$

c_k →


Approx. Kernel K-means

Approximate Kernel k-means: Solution to Large Scale Kernel Clustering, KDD 2011



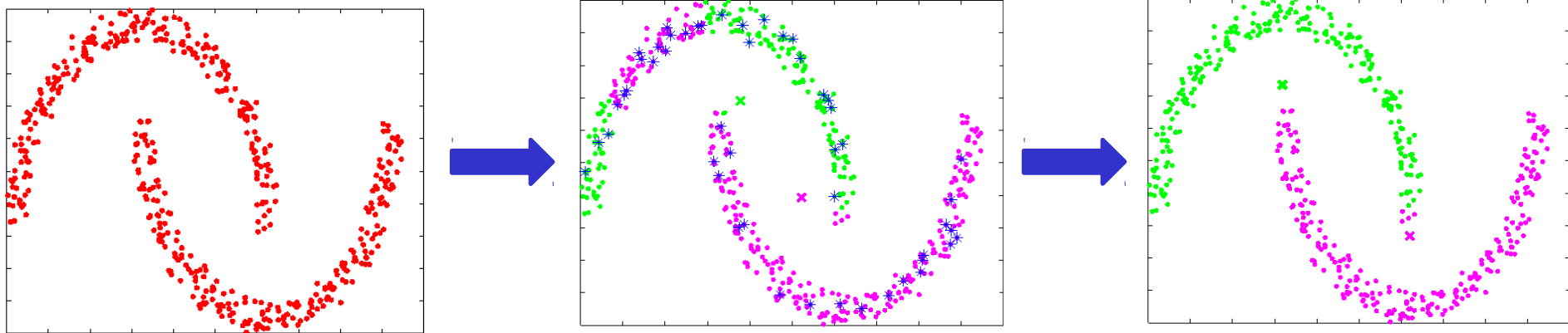
Iteratively optimize for the cluster centers

$$\min_U \max_{\alpha} \sum_{k=1}^C \sum_{i=1}^n U_{ik} \left\| \varphi(x_i) - \sum_{j=1}^m \alpha_{jk} \varphi(y_j) \right\|$$

c_k 

Approx. Kernel K-means

Approximate Kernel k-means: Solution to Large Scale Kernel Clustering, KDD 2011



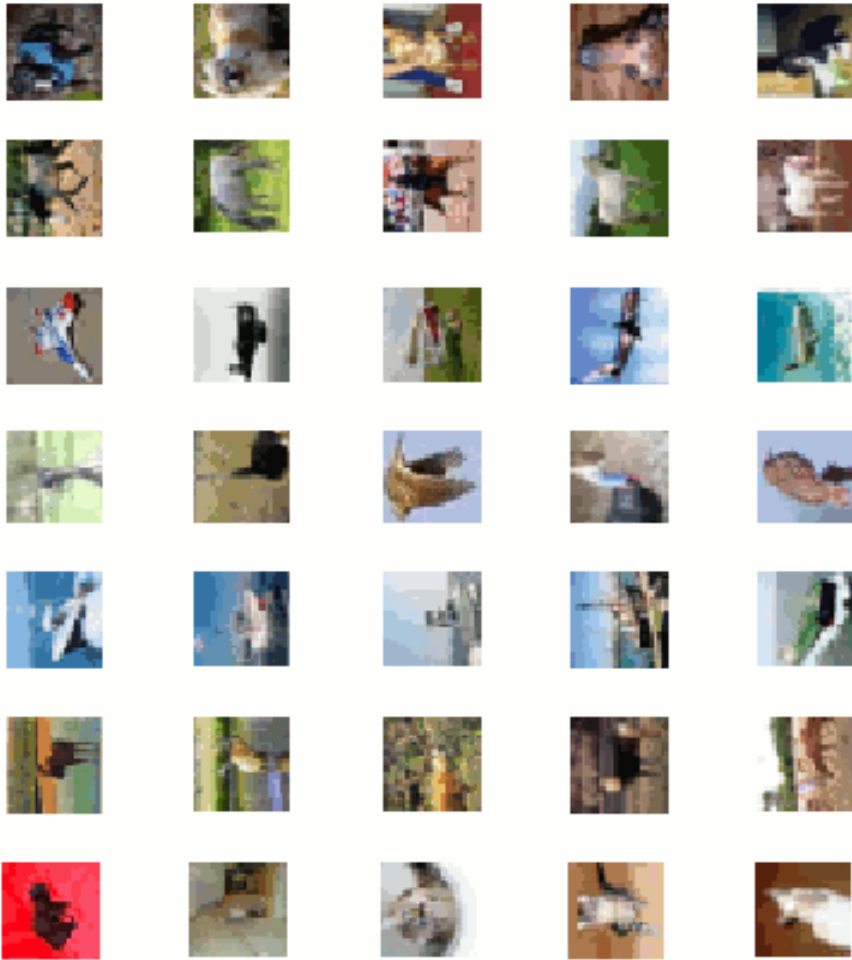
Obtain the final cluster labels after convergence

Equivalent to running K-means on $K_B K_A^{-1} K_B^T$ Nystrom approximation

Properties

- Need to calculate only $O(nm)$ sized matrix – almost linear runtime and memory complexity.
- Bounded $O(1/m)$ approximation error.

Clustering 80M Tiny Images



Example Clusters

Average clustering time into 100 clusters

Approximate kernel
K-means (m=1,000)

8.5 hours

K-means

6 hours

2.4 Ghz, 150 GB memory

Clustering 80M Tiny Images

Best Supervised Classification Accuracy on CIFAR-10 using GIST features: 54.7*

Clustering accuracy on CIFAR-10	
Kernel K-means	29.94
Approximate kernel K-means (m = 5,000)	29.76
Nystrom approximation based spectral clustering**	27.09
K-means	26.70

*Ranzato et. Al., Modeling pixel means and covariances using factorized third-order boltzmann machines, CVPR 2010

** Fowlkes et al., *Spectral grouping using the Nystrom method*, PAMI 2004

Matrix approximations

- Nystrom

- Randomly select columns

$$K = K_B K_A^{-1} K_B^T$$

- CUR

- Special form of Nystrom for any matrix K
- Select the most important columns C and rows R
- Find U that minimizes the approximation error

$$\|K - CUR\|_F$$

- Sparse approximation

- Find matrix N such that $\hat{K} = K + N$ is sparse.
- $E[N_{ij}] = 0$ and $var(N_{ij})$ is small

Out-of-sample clustering

To cluster a new point x using kernel K-means

$$d^2(x, c_k) = \kappa(x, x) - \frac{2}{n_k} \sum_{j=1}^n U_{kj} \kappa(x, x_j) + \frac{1}{n_k^2} \sum_{j=1}^n \sum_{l=1}^n U_{kj} U_{kl} \kappa(x_j, x_l)$$

Out-of-sample clustering

To cluster a new point x using kernel K-means

$$d^2(x, c_k) = \kappa(x, x) - \frac{2}{n_k} \sum_{j=1}^n U_{kj} \kappa(x, x_j) + \frac{1}{n_k^2} \sum_{j=1}^n \sum_{l=1}^n U_{kj} U_{kl} \kappa(x_j, x_l)$$

↑
Compute $O(n)$ kernel values

↑
Store the full kernel matrix

Out-of-sample clustering

To cluster a new point x using kernel K-means

$$d^2(x, c_k) = \kappa(x, x) - \frac{2}{n_k} \sum_{j=1}^n U_{kj} \kappa(x, x_j) + \frac{1}{n_k^2} \sum_{j=1}^n \sum_{l=1}^n U_{kj} U_{kl} \kappa(x_j, x_l)$$

↑
Compute $O(n)$ kernel values

↑
Store the full kernel matrix

No explicit representation for the centers

Out-of-sample clustering

- Kernel PCA
 - Project the data onto the first C eigenvectors (principal components) of the kernel matrix.
 - Perform clustering in the eigenspace.
 - Ref: Multiway Spectral Clustering with Out-of-Sample Extensions through Weighted Kernel PCA, Alzate et. al., PAMI 2010
- **Non-linear random projections**

Non-linear Random Projections

Efficient Kernel Clustering Using Random Fourier Features, ICDM 2012

- Random Projection for dimensionality reduction

- Generate a random $d \times r$ matrix R .
- Project data into r -dimensional space

$$x' = \frac{1}{\sqrt{r}} R^T x \quad y' = \frac{1}{\sqrt{r}} R^T y$$

- **Johnson-Lindenstrauss Lemma**

If R is orthonormal and r is sufficiently large, distances are preserved in the projected space.

$$(1 - \epsilon) \|x - y\|^2 \leq \|x' - y'\|^2 \leq (1 + \epsilon) \|x - y\|^2$$

Non-linear Random Projections

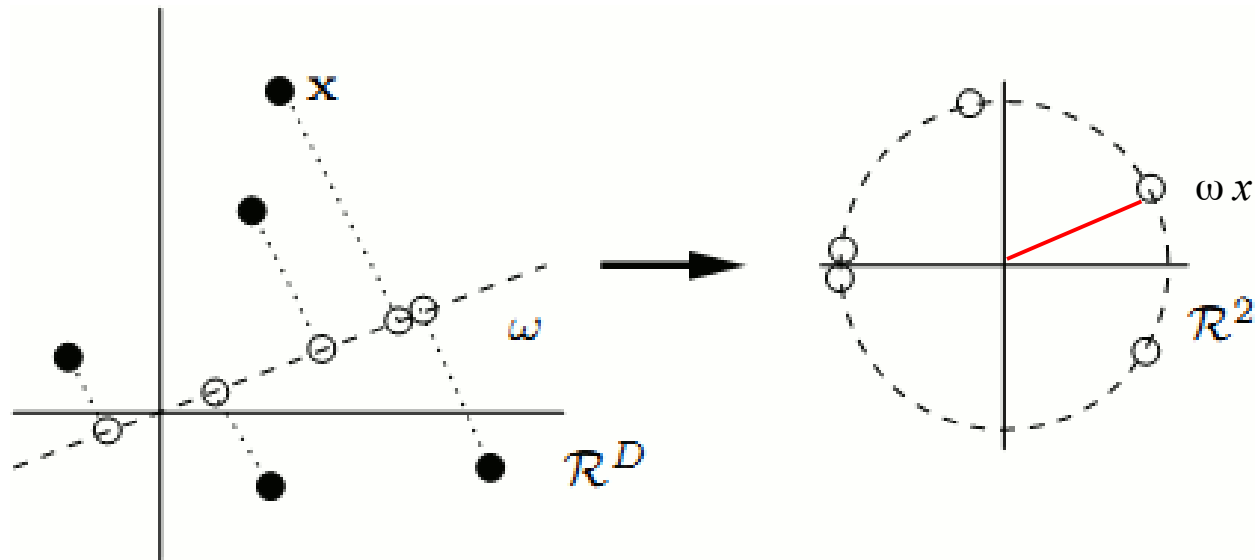
Efficient Kernel Clustering Using Random Fourier Features, ICDM 2012

- Kernel K-means is K-means in Hilbert space where data is linearly separable.
- **Efficiently** project data in Hilbert space to a low-dimensional space.
- Linear separability is preserved.
- Apply K-means in the low-dimensional space.
- Obtain low-dimensional representation of cluster centers.

Non-linear Random Projections

Efficient Kernel Clustering Using Random Fourier Features, ICDM 2012

Random Fourier Features



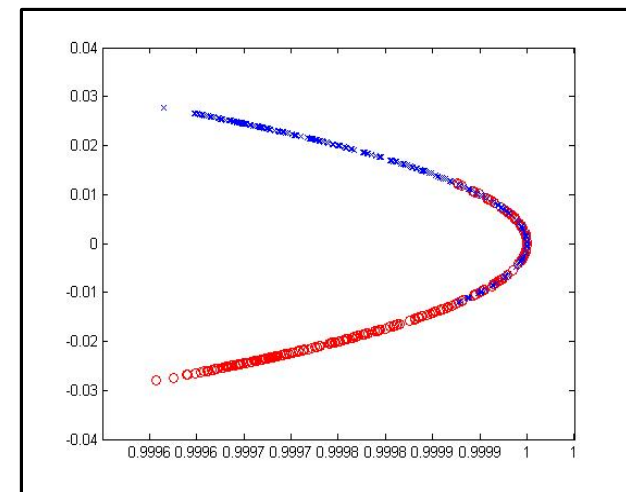
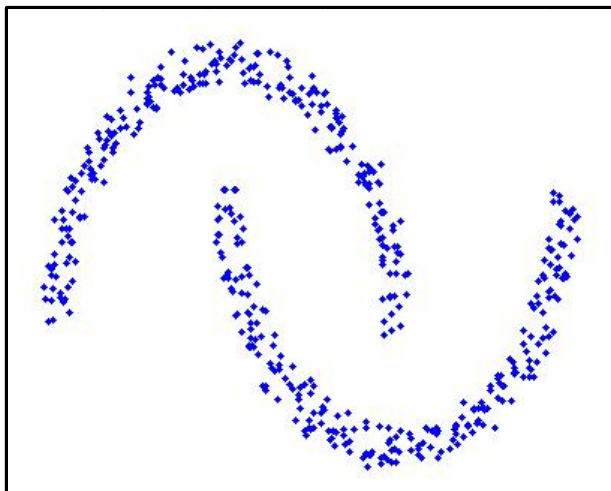
Map data into low-dimensional space using $f(\omega, x)$

$$f(\omega, x) = [\cos(\omega^T x) \sin(\omega^T x)]^T$$

$$\kappa(x, y) = E_{\omega} [f(\omega, x)^T f(\omega, y)]$$

Non-linear Random Projections

Efficient Kernel Clustering Using Random Fourier Features, ICDM 2012



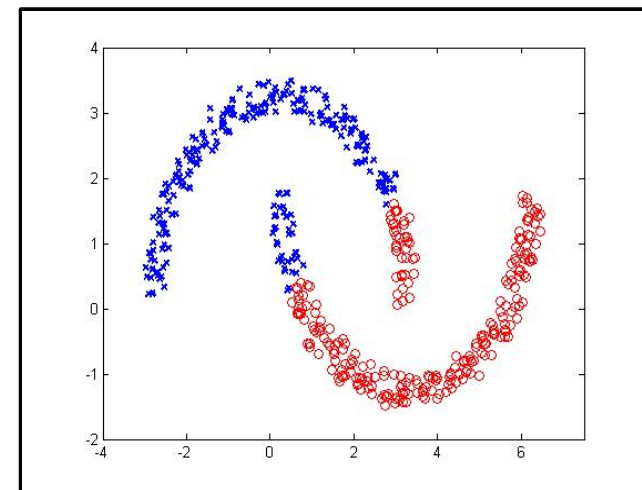
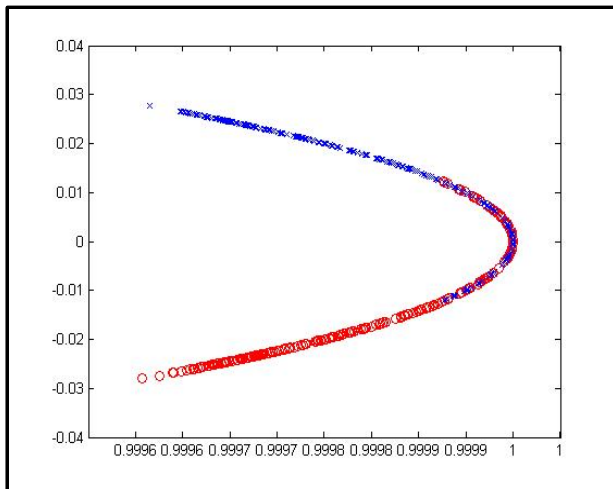
Randomly sample m vectors $\{\omega_1, \omega_2, \dots, \omega_m\}$, $m \ll n$ from Fourier transform of the kernel and project the points using $f(\omega, x)$

$$z(x) = \frac{1}{\sqrt{m}} \left[\cos(\omega_1^T x) \dots \cos(\omega_m^T x) \dots \sin(\omega_1^T x) \sin(\omega_m^T x) \right]$$

$$H = \begin{bmatrix} z(x_1)^T & z(x_2)^T & \dots & z(x_n)^T \end{bmatrix}$$

Non-linear Random Projections

Efficient Kernel Clustering Using Random Fourier Features, ICDM 2012



Cluster H using K-means and obtain the partition

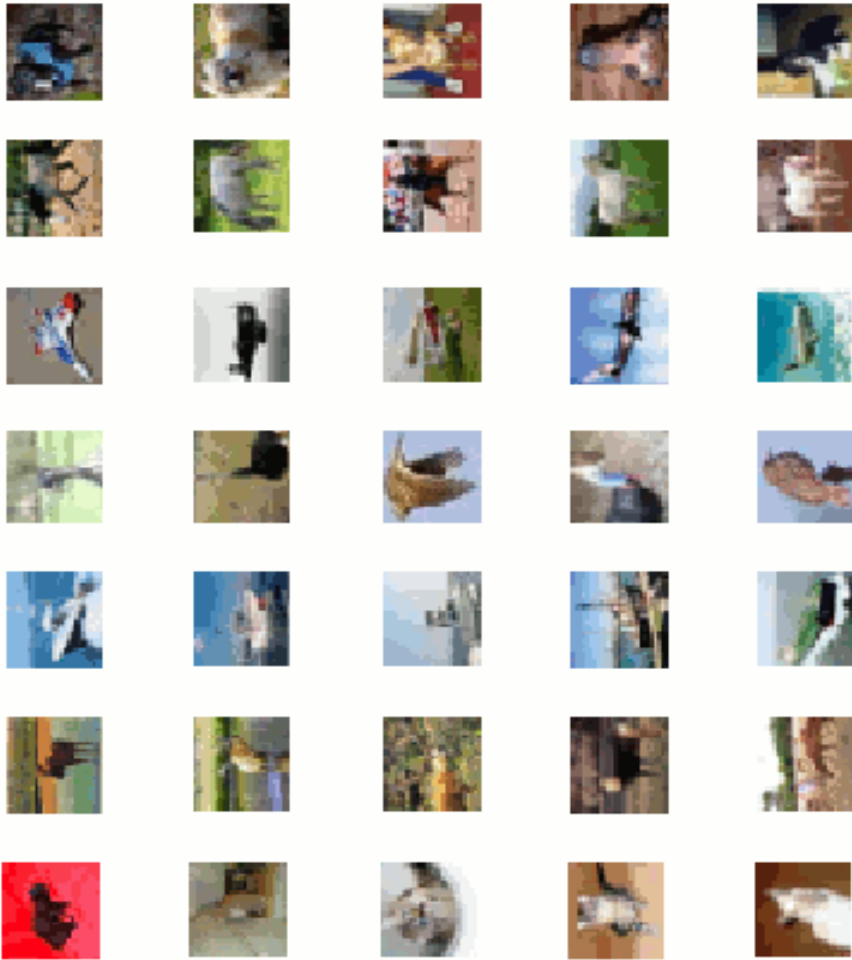
Properties

→ $O(nmd)$ runtime complexity and $O(nm)$ memory complexity.

→ Bounded $O(1/\sqrt{m})$ approximation error.

→ Explicit representation for the centers, add a new point by projecting and assigning to the closest center.

Clustering 80M Tiny Images



Example Clusters

Average clustering time into 100 clusters

Approximate kernel
K-means (m=1,000)

8.5 hours

Clustering using Random
Fourier Features (m=1,000)

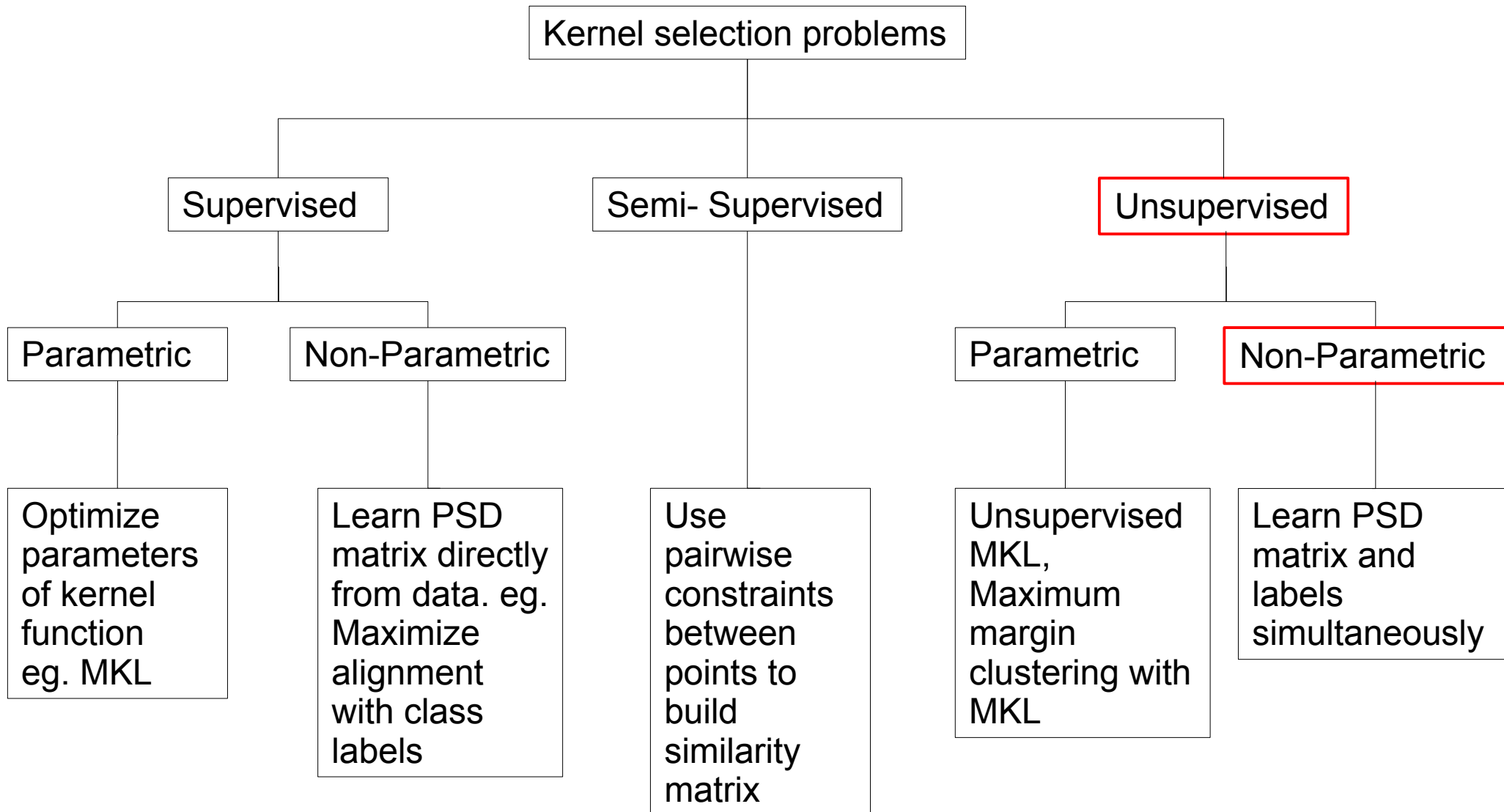
7.8 hours

K-means

6 hours

2.4 Ghz, 150 GB memory

Kernel Selection



Unsupervised Non-Parametric Kernel Learning

Unsupervised non-parametric kernel learning algorithm, Liu et. al., 2013

- Highly flexible, least amount of prior knowledge required
- Learn kernel and labels simultaneously
- Maintain the spectrum of the data after projection into feature space

$$\begin{aligned} \min_{K, U} \text{trace}(KL) + \frac{1}{2} U \left(K + \frac{\gamma}{C} KL + \frac{I}{C} \right)^{-1} U^T \\ \text{s.t.} \\ K \succeq 0 \\ \text{trace}(K^p) \leq B \\ -l \leq n_p - n_q \leq l \\ L = I - D^{-1/2} W D^{-1/2} \end{aligned}$$

Unsupervised Non-Parametric Kernel Learning

Unsupervised non-parametric kernel learning algorithm, Liu et. al., 2013

- Highly flexible, least amount of prior knowledge required
- Learn kernel and labels simultaneously
- Maintain the spectrum of the data after projection into feature space

Spectrum regularization

$$\min_{K, U} \text{trace}(KL) + \frac{1}{2} U \left(K + \frac{\gamma}{C} KL + \frac{I}{C} \right)^{-1} U^T$$

s.t.

$$K \succeq 0$$

$$\text{trace}(K^p) \leq B$$

$$-l \leq n_p - n_q \leq l$$

$$L = I - D^{-1/2} W D^{-1/2}$$

Graph Laplacian

Unsupervised Non-Parametric Kernel Learning

Unsupervised non-parametric kernel learning algorithm, Liu et. al., 2013

- Highly flexible, least amount of prior knowledge required
- Learn kernel and labels simultaneously
- Maintain the spectrum of the data after projection into feature space

Squared loss

$$\min_{K, U} \text{trace}(KL) + \frac{1}{2} U \left(K + \frac{\gamma}{C} KL + \frac{I}{C} \right)^{-1} U^T$$

s.t.

$$K \succeq 0$$

$$\text{trace}(K^p) \leq B$$

$$-l \leq n_p - n_q \leq l$$

$$L = I - D^{-1/2} W D^{-1/2}$$

Unsupervised Non-Parametric Kernel Learning

Unsupervised non-parametric kernel learning algorithm, Liu et. al., 2013

- Highly flexible, least amount of prior knowledge required
- Learn kernel and labels simultaneously
- Maintain the spectrum of the data after projection into feature space

$$\min_{K, U} \text{trace}(KL) + \frac{1}{2} U \left(K + \frac{\gamma}{C} KL + \frac{I}{C} \right)^{-1} U^T$$

s.t.

$$K \succeq 0 \quad \text{PSD}$$

$$\text{trace}(K^p) \leq B$$

$$-l \leq n_p - n_q \leq l$$

$$L = I - D^{-1/2} W D^{-1/2}$$

Unsupervised Non-Parametric Kernel Learning

Unsupervised non-parametric kernel learning algorithm, Liu et. al., 2013

- Highly flexible, least amount of prior knowledge required
- Learn kernel and labels simultaneously
- Maintain the spectrum of the data after projection into feature space

$$\min_{K, U} \text{trace}(KL) + \frac{1}{2} U \left(K + \frac{\gamma}{C} KL + \frac{I}{C} \right)^{-1} U^T$$

s.t.

$$K \succeq 0$$

Avoid overfitting and assigning all points to one cluster

$$\text{trace}(K^p) \leq B$$
$$-l \leq n_p - n_q \leq l$$

$$L = I - D^{-1/2} W D^{-1/2}$$

Unsupervised Non-Parametric Kernel Learning

Unsupervised non-parametric kernel learning algorithm, Liu et. al., 2013

- Highly flexible, least amount of prior knowledge required
- Learn kernel and labels simultaneously
- Maintain the spectrum of the data after projection into feature space

$$\begin{aligned} \min_{K, U} \text{trace}(KL) + \frac{1}{2} U \left(K + \frac{\gamma}{C} KL + \frac{I}{C} \right)^{-1} U^T \\ \text{s.t.} \\ K \succeq 0 \\ \text{trace}(K^p) \leq B \\ -l \leq n_p - n_q \leq l \\ L = I - D^{-1/2} W D^{-1/2} \end{aligned}$$

Solution involves using SDP, eigendecomposition – $O(n^2)$ to $O(n^3)$ complexity

Summary

- Kernel clustering more accurate than linear clustering
- Challenges: Non-scalability, out-of-sample clustering and kernel selection
- Approximation and parallelization techniques to handle scalability
- Projections for out-of-sample clustering
- Unsupervised kernel learning is flexible but complex

References

Kernel Theory

- [1] Statistical Learning Theory, V.N. Vapnik
- [2] Learning with Kernels, B. Scholkopf and A. Smola

Matrix Approximation

- [3] On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-Based Learning, Drineas and Mahoney
- [4] CUR matrix decompositions for improved data analysis, Drineas and Mahoney
- [5] Improving CUR Matrix Decomposition and the Nystrom Approximation via Adaptive Sampling, Wang and Zhang
- [6] Fast Computation of Low Rank Matrix Approximations, Achlioptas and McSherry

References

Random Projections

[7] Kernels as Features: On kernels, margins, and low-dimensional mappings, Balcan, Blum and Vempala

[8] Random Features for Large-Scale Kernel Machines, Rahimi and Recht

Kernel Learning

[9] Generalized Maximum Margin Clustering and Unsupervised Kernel Learning, Valizadegan and Jin