A Tour of Unsupervised Learning – Part II
Clustering. Semi-supervised, active and reinforcement learning

Marina Meilă
mmp@stat.washington.edu

Department of Statistics
University of Washington

MPS2016

Outline

Outline

What is clustering? Problem and Notation

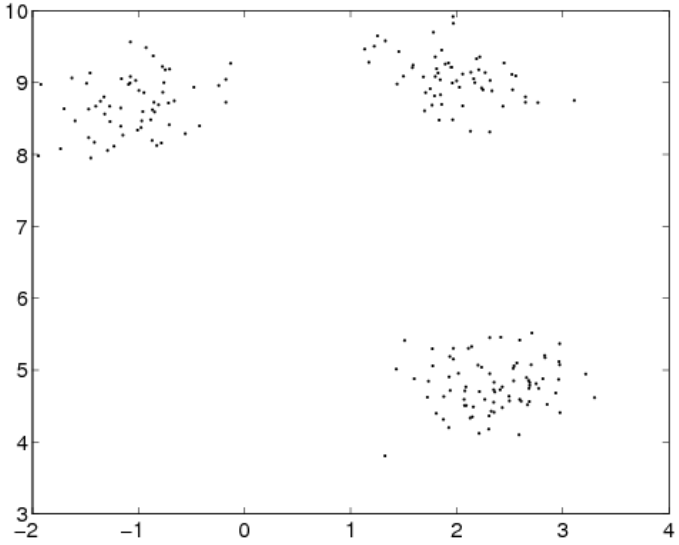
- **Informal definition** Clustering = Finding groups in data
- **Notation**

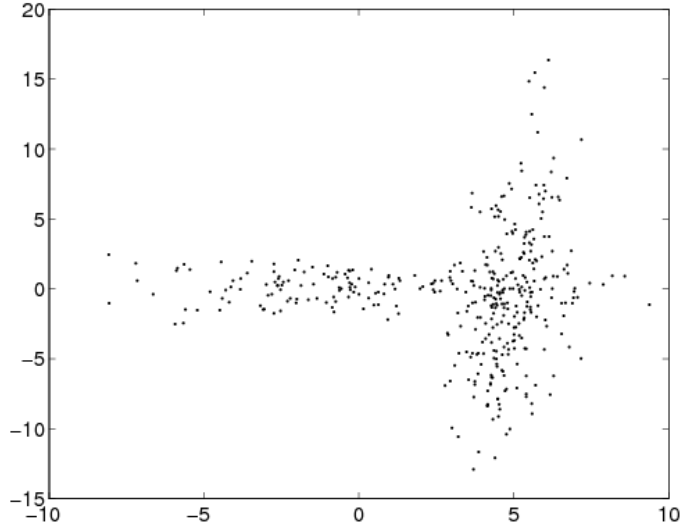
$\mathcal{D}$	$=$	$\{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}$ a data set
n	$=$	number of data points
K	$=$	number of clusters ($K << n$)
Δ	$=$	$\{C_1, C_2, \ldots, C_K\}$ a partition of $\mathcal{D}$ into disjoint subsets
$k(i)$	$=$	the label of point i
$\mathcal{L}(\Delta)$	$=$	cost (loss) of Δ (to be minimized)

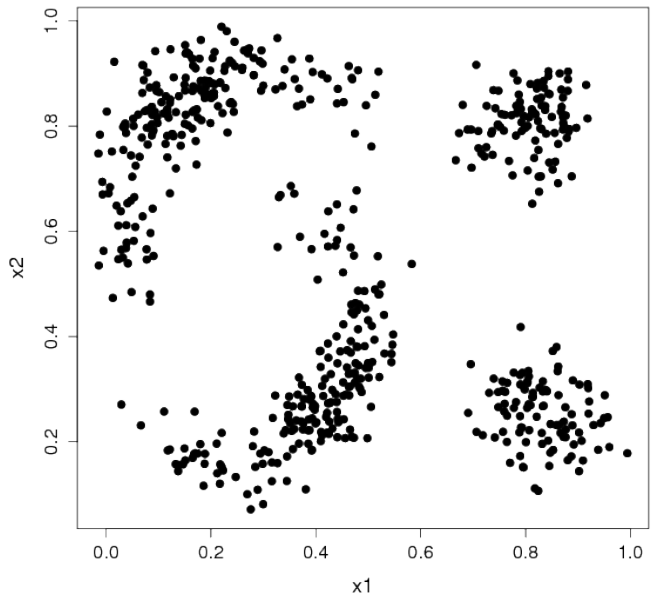
- **Second informal definition** Clustering = given n data points, separate them into K clusters
- Hard vs. soft clusterings
  - Hard clustering Δ: an item belongs to only 1 cluster
  - Soft clustering $\gamma = \{\gamma_{ki}\}_{k=1:K}^{i=1:n}$
    γ_{ki} = the degree of membership of point i to cluster k

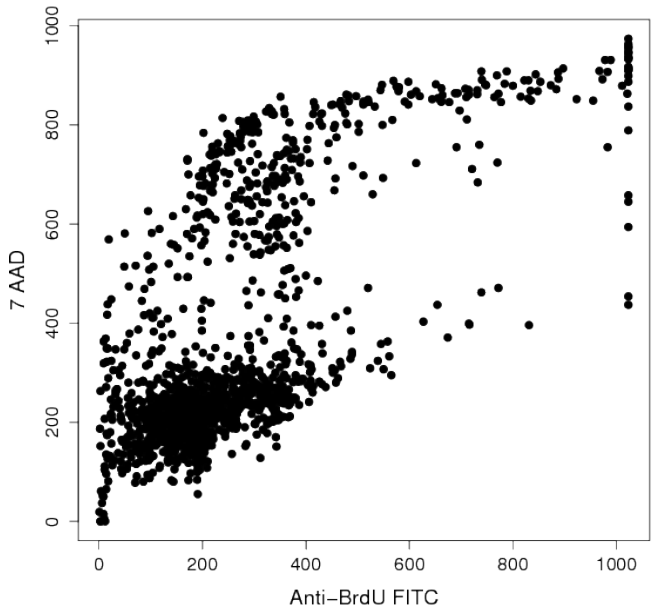
$$\sum_k \gamma_{ki} = 1 \quad \text{for all } i$$

(usually associated with a probabilistic model)

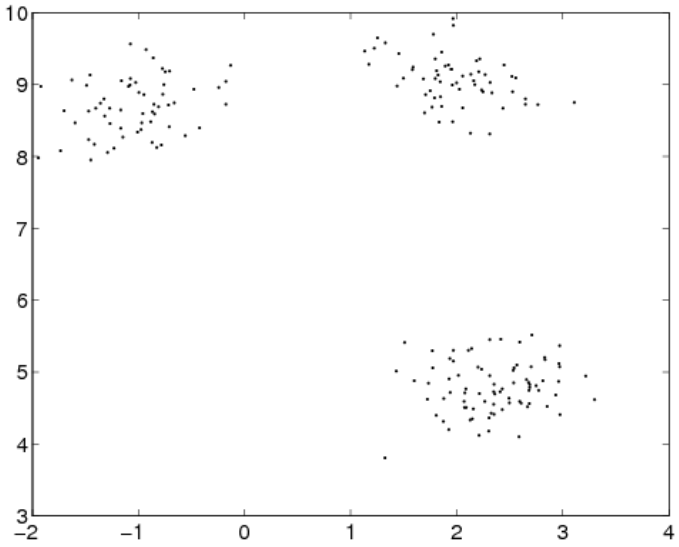

10
9
8
7
6
5
4
3
−2
−1
0
1
2
3
4

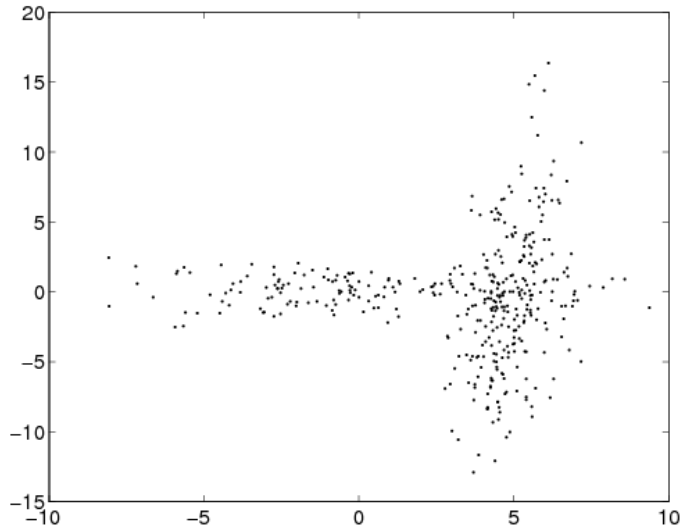

20
15
10
5
0
−5
−10
−15
−10
−5
0
5
10

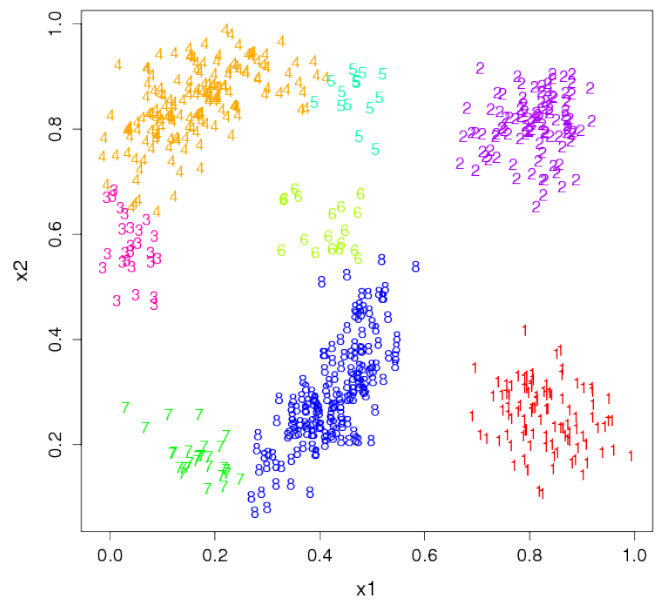

x2
1.0
0.8
0.6
0.4
0.2
0.0
0.2
0.4
0.6
0.8
1.0
x1

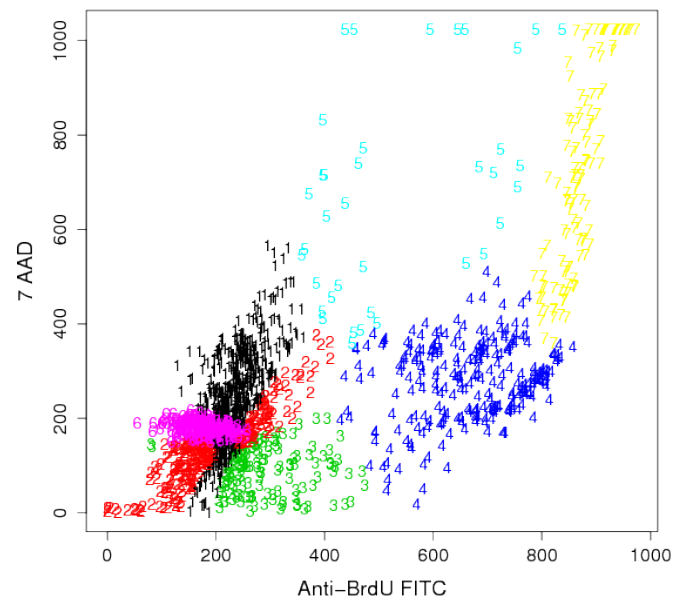

7 AAD
1000
800
600
400
200
0
0
200
400
600
800
1000
Anti−BrdU FITC

(from)

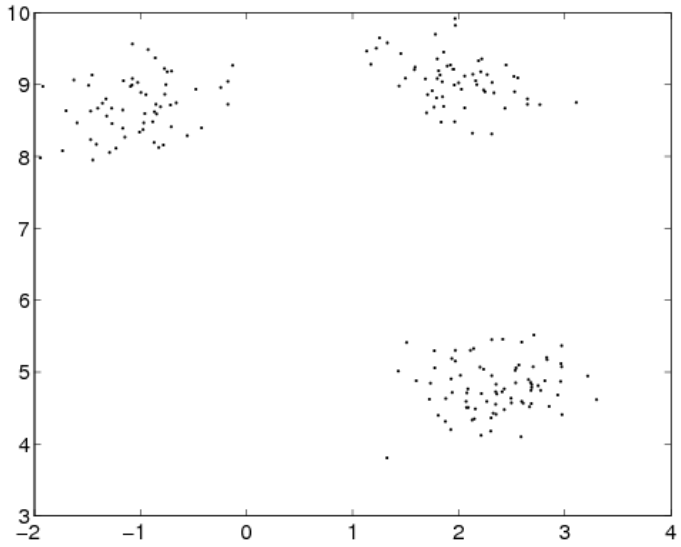
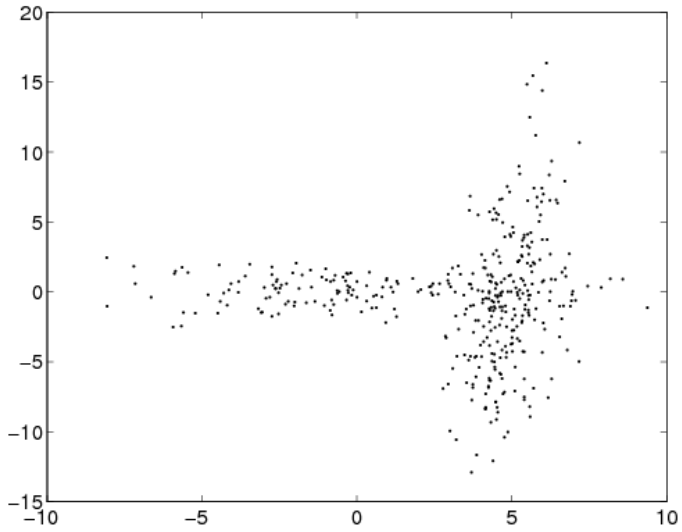
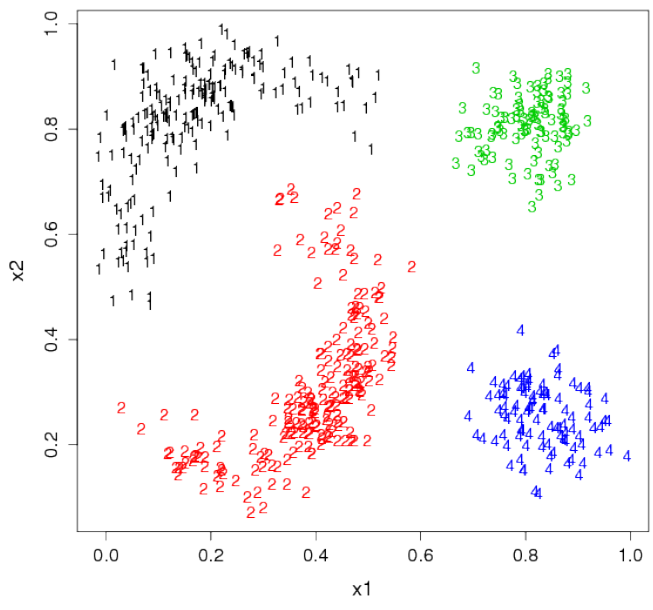


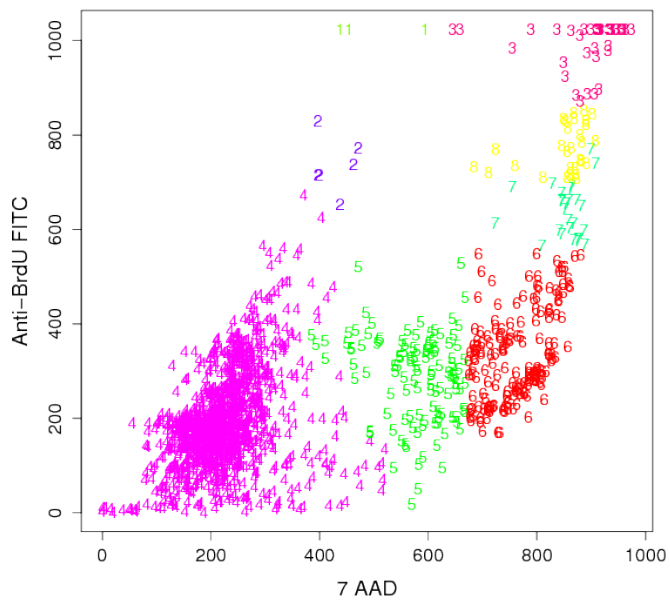






(from)



(from)

Paradigms

Depend on type of data, type of clustering, type of cost (probabilistic or not), and constraints (about K, shape of clusters)

- Data = vectors $\{x_i\}$ in $\mathbb{R}^d$

Parametric	Cost based [hard]
(K known)	Model based [soft]
Non-parametric	Dirichlet process mixtures [soft]
(K determined	Information bottleneck [soft]
by algorithm)	Modes of distribution [hard]
	Gaussian blurring mean shift [hard]

- Data = similarities between pairs of points $[S_{ij}]_{i,j=1:n}$, $S_{ij} = S_{ji} \geq 0$ (called Similarity based clustering)

Graph partitioning	spectral clustering [hard, K fixed, cost based]
	typical cuts [hard non-parametric, cost based]
Affinity propagation	[hard/soft non-parametric]

Classification vs Clustering

	Classification	**Clustering**
Performance criterion	Expected error	a wide variety
	Supervised	Unsupervised
Generalization	Performance on new data is what matters	Performance on current data is what matters
K	Known	Unknown
"Goal"	Prediction	Exploration, etc
Stage of field	Mature	Young: new paradigms and theoretical results are emerging

Outline

K-means clustering

Algorithm K-Means

Input Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters K

Initialize centers $\mu_1, \mu_2, \ldots \mu_K \in \mathbb{R}^d$ at random

Iterate until convergence

1. for $i = 1 : n$ (assign points to clusters $\Rightarrow$ new clustering)

$$k(i) = \operatorname*{argmin}_k ||x_i - \mu_k||$$

2. for $k = 1 : K$ (recalculate centers)

$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i \tag{1}$$

K-means clustering

Algorithm K-Means

Input Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters K

Initialize centers $\mu_1, \mu_2, \ldots \mu_K \in \mathbb{R}^d$ at random

Iterate until convergence

1. for $i = 1 : n$ (assign points to clusters $\Rightarrow$ new clustering)

$$k(i) = \operatorname*{argmin}_k ||x_i - \mu_k||$$

2. for $k = 1 : K$ (recalculate centers)

$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i \tag{1}$$

- Convergence
  - if Δ doesn't change at iteration m it will never change after that
  - convergence in finite number of steps

K-means clustering

Algorithm K-Means

Input Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters K

Initialize centers $\mu_1, \mu_2, \ldots \mu_K \in \mathbb{R}^d$ at random

Iterate until convergence

1. for $i = 1 : n$ (assign points to clusters $\Rightarrow$ new clustering)

$$k(i) = \operatorname*{argmin}_k ||x_i - \mu_k||$$

2. for $k = 1 : K$ (recalculate centers)

$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i \tag{1}$$

- Convergence
  - if Δ doesn't change at iteration m it will never change after that
  - convergence in finite number of steps to local optimum of cost $\mathcal{L}$ (defined next)

K-means clustering

Algorithm K-Means

Input Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters K

Initialize centers $\mu_1, \mu_2, \ldots \mu_K \in \mathbb{R}^d$ at random

Iterate until convergence

1. for $i = 1 : n$ (assign points to clusters $\Rightarrow$ new clustering)

$$k(i) = \operatorname*{argmin}_k ||x_i - \mu_k||$$

2. for $k = 1 : K$ (recalculate centers)

$$\mu_k = \frac{1}{|C_k|} \sum_{i \in C_k} x_i \tag{1}$$

- Convergence
  - if Δ doesn't change at iteration m it will never change after that
  - convergence in finite number of steps to local optimum of cost $\mathcal{L}$ (defined next)
  - therefore, initialization will matter

The K-means cost

$$\mathcal{L}(\Delta) = \sum_{k=1}^{K} \sum_{i \in C_k} ||x_i - \mu_k||^2$$

- K-means solves a least-squares problem
- the cost $\mathcal{L}$ is called quadratic distortion

Proposition The K-means algorithm decreases $\mathcal{L}(\Delta)$ at every step.

Sketch of proof

- step 1: reassigning the labels can only decrease $\mathcal{L}$
- step 2: reassigning the centers μ_k can only decrease $\mathcal{L}$ because μ_k as given by (1) is the solution to

$$\mu_k = \min_{\mu \in \mathbb{R}^d} \sum_{i \in C_k} ||x_i - \mu||^2 \tag{2}$$

Initialization of the centroids $\mu_{1:K}$

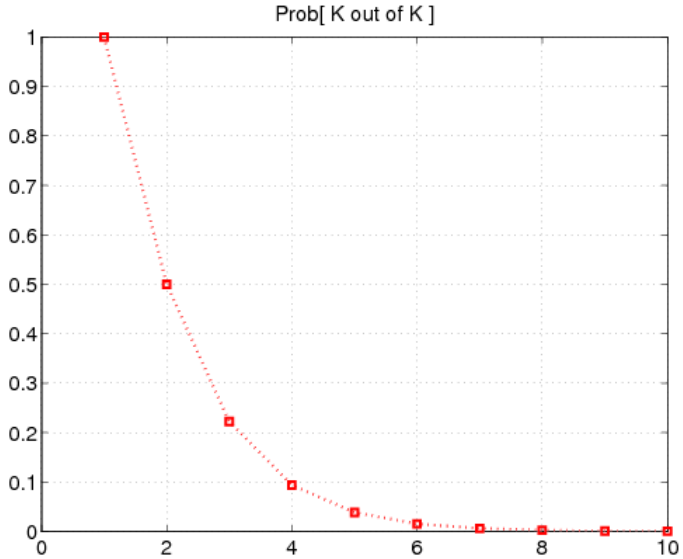
- Idea 1: start with K points at random

Initialization of the centroids $\mu_{1:K}$

- Idea 1: start with K points at random
- Idea 2: start with K data points at random

Initialization of the centroids $\mu_{1:K}$

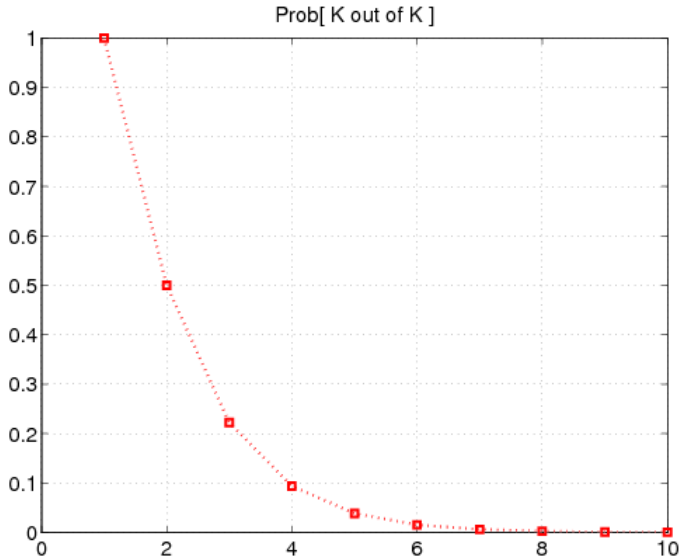
- Idea 1: start with K points at random
- Idea 2: start with K data points at random
  What's wrong with chosing K data points at random?



The probability of hitting all K clusters with K samples approaches 0 when $K > 5$

Initialization of the centroids $\mu_{1:K}$

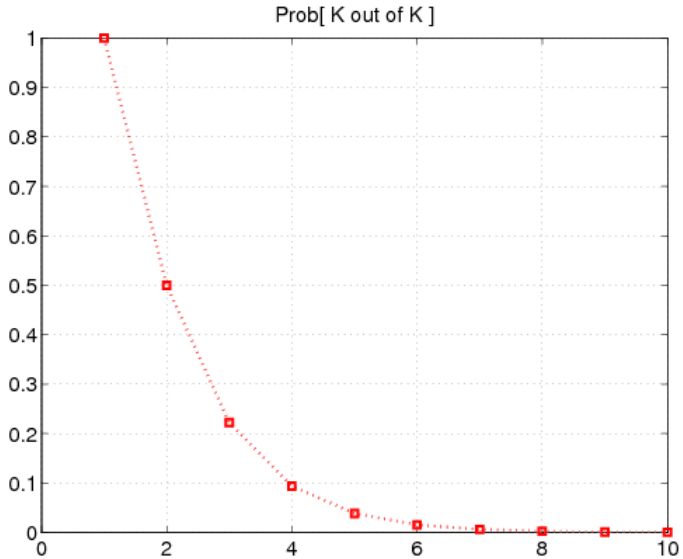
- Idea 1: start with K points at random
- Idea 2: start with K data points at random
  What's wrong with chosing K data points at random?



  The probability of hitting all K clusters with K samples approaches 0 when $K > 5$
- Idea 3: start with K data points using Fastest First Traversal (greedy simple approach to spread out centers)

Initialization of the centroids $\mu_{1:K}$

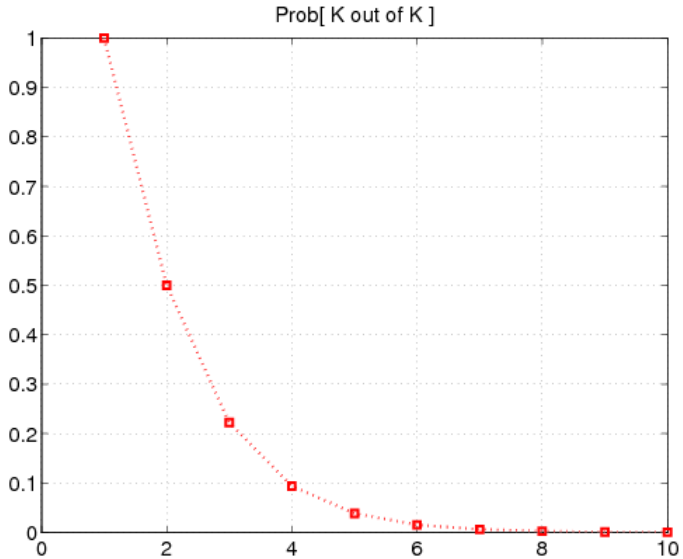
- Idea 1: start with K points at random
- Idea 2: start with K data points at random
  What's wrong with chosing K data points at random?



  The probability of hitting all K clusters with K samples approaches 0 when $K > 5$
- Idea 3: start with K data points using Fastest First Traversal (greedy simple approach to spread out centers)
- Idea 4: k-means++ (randomized, theoretically backed approach to spread out centers)

Initialization of the centroids $\mu_{1:K}$

- Idea 1: start with K points at random
- Idea 2: start with K data points at random
  What's wrong with chosing K data points at random?



  The probability of hitting all K clusters with K samples approaches 0 when $K > 5$
- Idea 3: start with K data points using Fastest First Traversal (greedy simple approach to spread out centers)
- Idea 4: k-means++ (randomized, theoretically backed approach to spread out centers)
- Idea 5: "K-logK" Initialization (start with enough centers to hit all clusters, then prune down to K)
  For EM Algorithm , for K-means

The "K-logK" initialization

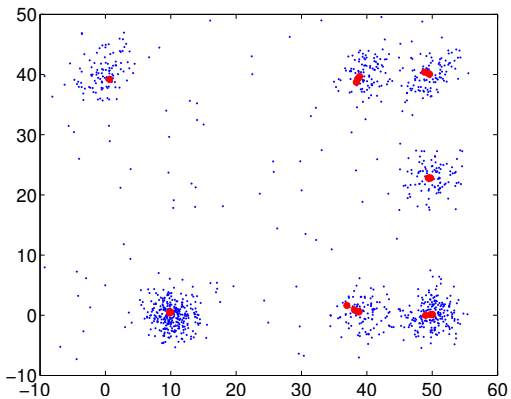
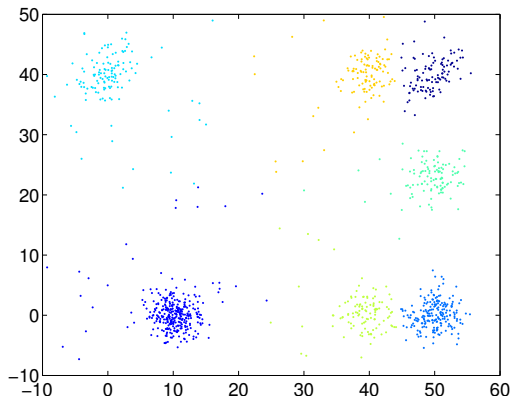
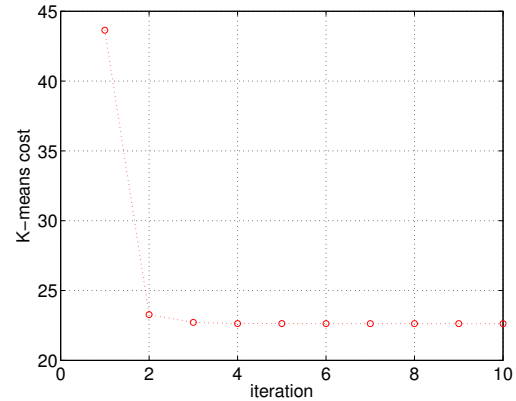
The K-logK Initialization (see also)

1. pick $\mu^0_{1:K'}$ at random from data set, where $K' = O(K \log K)$
   (this assures that each cluster has at least 1 center w.h.p)
2. run 1 step of K-means
3. remove all centers μ^0_k that have few points, e.g $|C_k| < \frac{n}{eK'}$
4. from the remaining centers select K centers by **Fastest First Traversal**
   - 4.1 pick μ_1 at random from the remaining $\{\mu^0_{1:K'}\}$
   - 4.2 for $k = 2 : K$, $\mu_k \leftarrow \underset{\mu^0_{k'}}{\operatorname{argmax}} \min_{j=1:k-1} ||\mu^0_{k'} - \mu_j||$, i.e next μ_k is furthest away from the already chosen centers
5. continue with the standard **K-means** algorithm

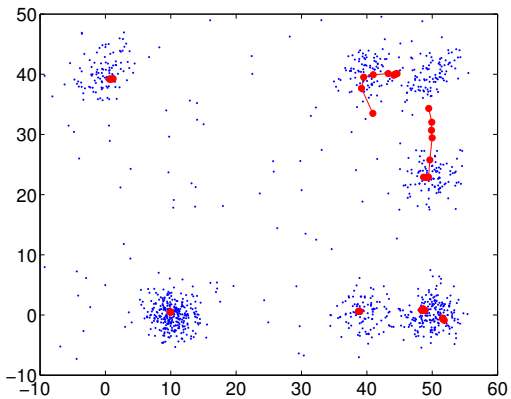
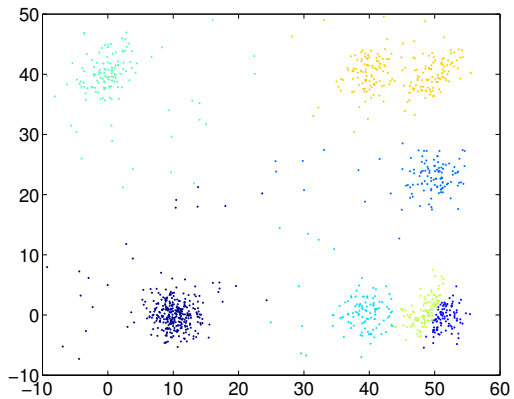
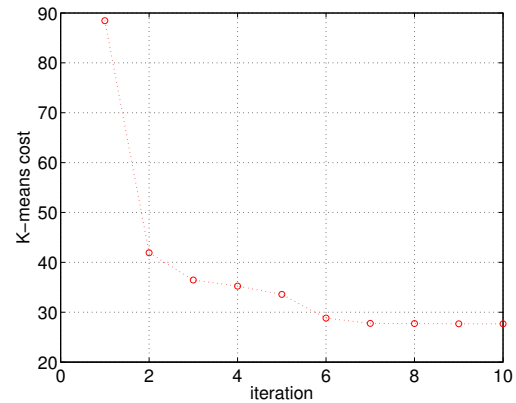
K-means clustering with K-logK Initialization

Example using a mixture of 7 Normal distributions with 100 outliers sampled uniformly

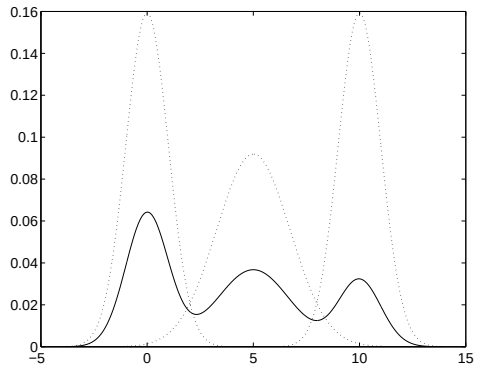
K-LOGK $K = 7,\ T = 100,\ n = 1100,\ c = 1$

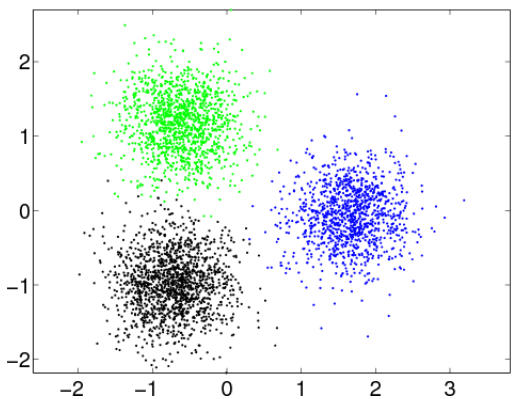




NAIVE $K = 7\ \ T = 100,\ n = 1100$

Model based clustering: Mixture models



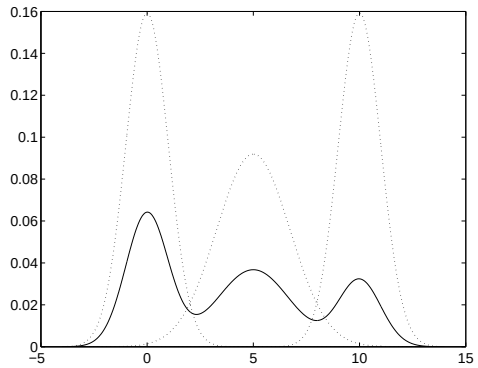


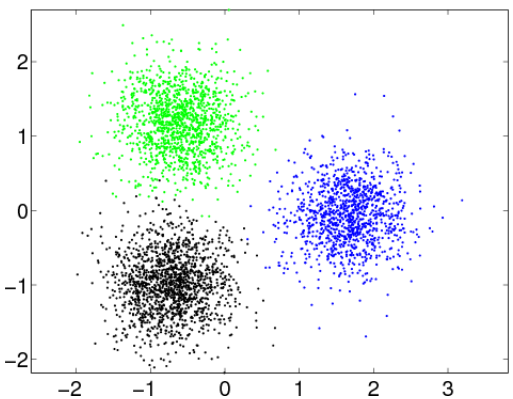
- The mixture density

$$f(x) = \sum_{k=1}^{K} \pi_k f_k(x)$$

- $f_k(x)$ = the components of the mixture
  - each is a density
  - f called mixture of Gaussians if $f_k = Normal_{\mu_k, \Sigma_k}$
- π_k = the mixing proportions, $\sum_k = 1^K \pi_k = 1, \quad \pi_k \geq 0.$
- model parameters $\theta = (\pi_{1:K}, \mu_{1:K}, \Sigma_{1:K})$

Model based clustering: Mixture models





- The mixture density

$$f(x) = \sum_{k=1}^{K} \pi_k f_k(x)$$

- $f_k(x) =$ the components of the mixture
  - each is a density
  - f called mixture of Gaussians if $f_k = Normal_{\mu_k, \Sigma_k}$
- $\pi_k =$ the mixing proportions, $\sum_k = 1^K \pi_k = 1, \quad \pi_k \geq 0.$
- model parameters $\theta = (\pi_{1:K}, \mu_{1:K}, \Sigma_{1:K})$
- The degree of membership of point i to cluster k

$$\gamma_{ki} \stackrel{\text{def}}{=} P[x_i \in C_k] = \frac{\pi_k f_k(x)}{f(x)} \quad \text{for } i = 1 : n, \; k = 1 : K \tag{5}$$

- depends on x_i and on the model parameters

The Expectation-Maximization (EM) Algorithm

Algorithm Expectation-Maximization (EM)

Input Data $\mathcal{D} = \{x_i\}_{i=1:n}$, number clusters K

Initialize parameters $\pi_{1:K} \in \mathbb{R}$, $\mu_{1:K} \in \mathbb{R}^d$, $\Sigma_{1:K} \in \mathbb{R}^{d \times d}$

Iterate until convergence

E step (Optimize clustering) for $i = 1 : n$, $k = 1 : K$

$$\text{compute } \gamma_{ki} \;=\; \frac{\pi_k f_k(x)}{f(x)}$$

M step (Optimize parameters)

- Compute "number of points" in cluster k

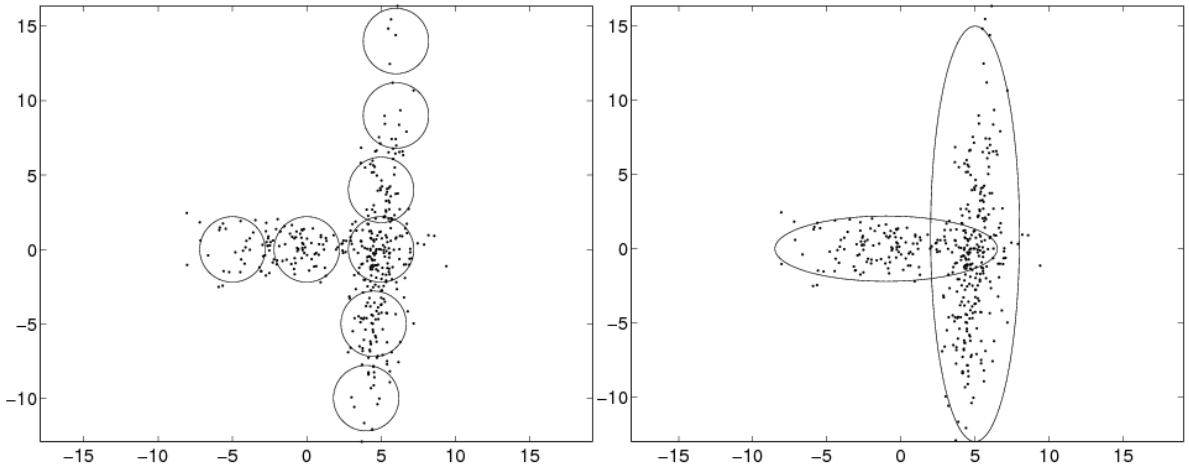
$$\Gamma_k \;=\; \sum_{i=1}^{m} \gamma_{ki}, \;\; k = 1 : K \qquad (\text{note: } \sum_k \Gamma_k \;=\; n) \tag{6}$$

- Estimate parameters

$$\begin{aligned} \pi_k &= \frac{\Gamma_k}{n}, \quad k = 1 : K \\ \mu_k &= \sum_{i=1}^{n} \frac{\gamma_{ki}}{\Gamma_k} x_i \\ \Sigma_k &= \frac{\sum_{i=1}^{n} \gamma_{ki} (x_i - \mu_k)(x_i - \mu_k)^T}{\Gamma_k} \end{aligned}$$

EM versus K-means

- Alternates between cluster assignments and parameter estimation
- Cluster assignments γ_{ki} are probabilistic
- Cluster parametrization more flexible



- Converges to local optimum of log-likelihood

  Initialization recommended by K-logK method

- Modern algorithms with guarantees (for e.g. mixtures of Gaussians)
  - Random projections
  - Projection on principal subspace
  - Two step EM (=K-logK initialization + one more EM iteration)

Outline

Similarity based clustering

- **Paradigm:** the features we observe are measures of similarity/dissimilarity between pairs of data points, e.g

	points	features
Image segmentation	pixels	distance in color space, location, separated by a contour, belong to same texture
Social network	people	friendship, coauthorship, hyperlinks, email
Text analysis	words	appear in same context

- The features are summarized by a single similarity measure S_{ij}
  - e.g $S_{ij} = e^{\sum_k \alpha_k \text{feature}_k(i,j)}$ for all points i, j
  - symmetric $S_{ij} = S_{ji}$
  - non-negative $S_{ij} \geq 0$

Similarity based clustering

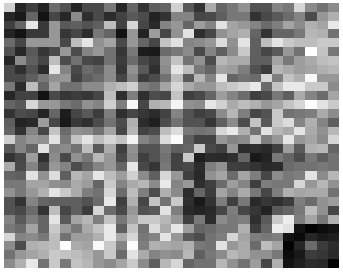
- **Paradigm:** the features we observe are measures of similarity/dissimilarity between pairs of data points, e.g

	points	features
Image segmentation	pixels	distance in color space, location, separated by a contour, belong to same texture
Social network	people	friendship, coauthorship, hyperlinks, email
Text analysis	words	appear in same context

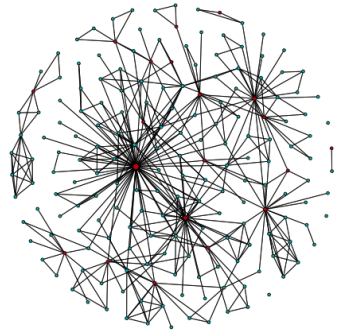
- The features are summarized by a single similarity measure S_{ij}
  - e.g $S_{ij} = e^{\sum_k \alpha_k \text{feature}_k(i,j)}$ for all points i, j
  - symmetric $S_{ij} = S_{ji}$
  - non-negative $S_{ij} \geq 0$

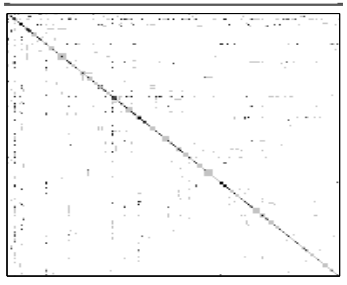
- Mathematically, we can see the data as
  - a $n \times n$ matrix $S = [S_{ij}]$
  - a (weighted) graph
    points = graph nodes, similarity S_{ij} = weight of edge ij
    meaningful because very few similarities are large
    Then, clustering is cutting the graph

Artificial matrix



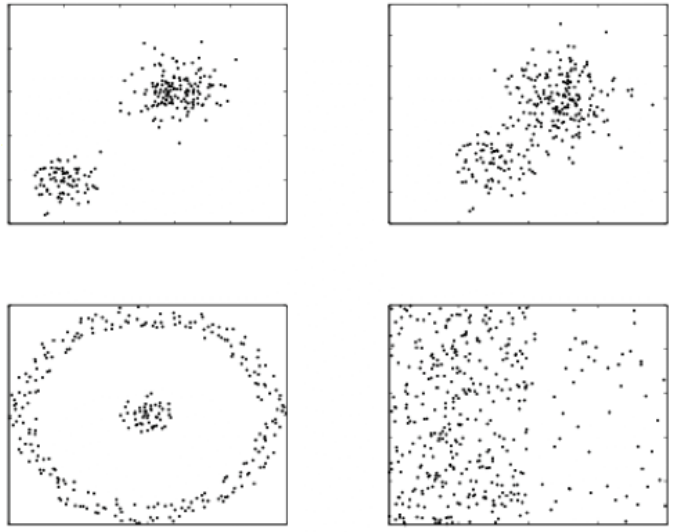
UW Statistics co-authorship data





$S_{ij} = \#$ tech-reports co-authored

Similarity based useful for $\mathbb{R}^d$ data as well



$$S_{ij} = e^{-\frac{||x_i - x_j||^2}{\sigma^2}}$$

Criteria for clustering

- Graph cuts

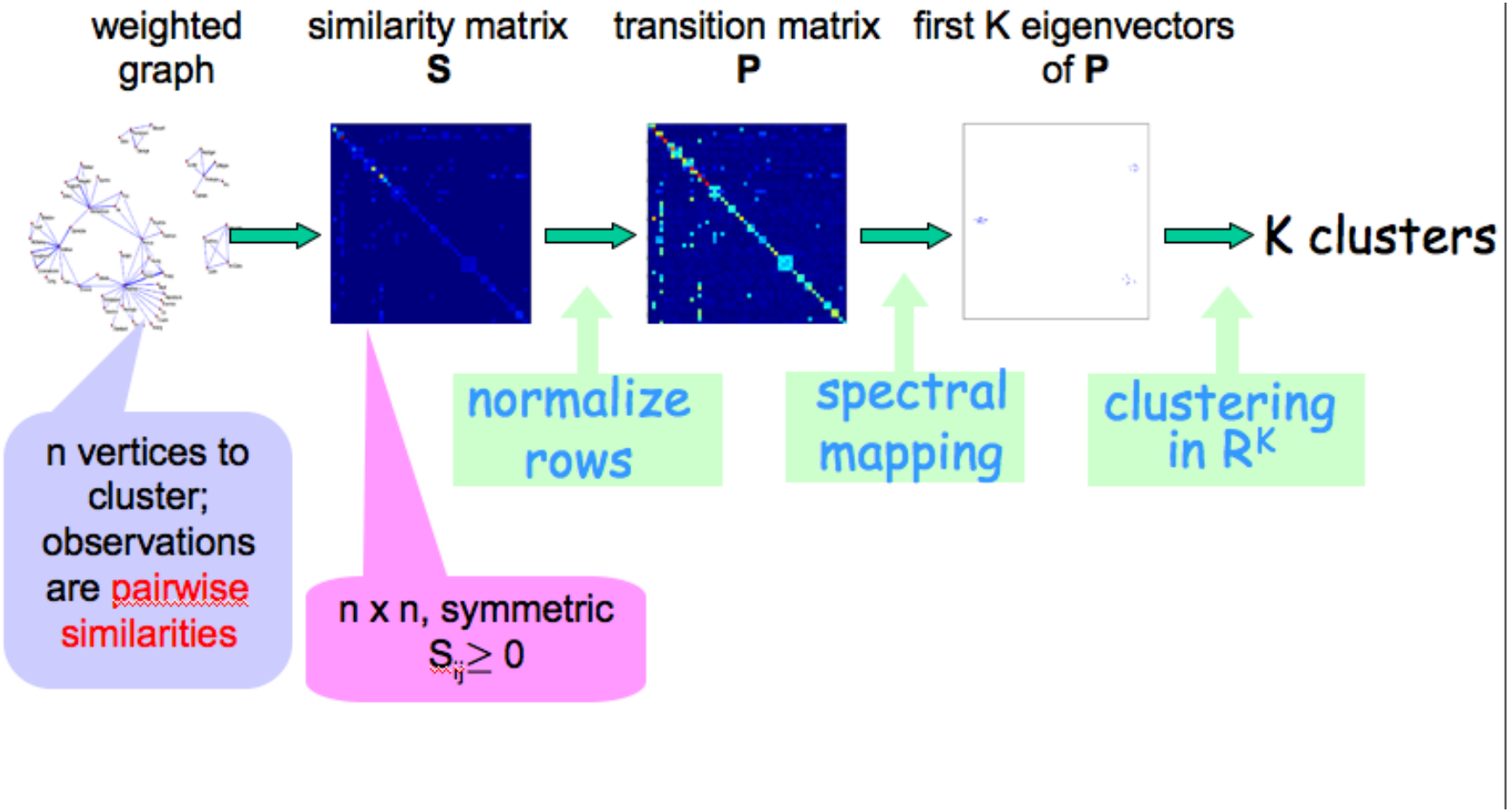
  remove some edges $\Longrightarrow$ disconnected graph

  the groups are the connected components
- By similar behavior

  nodes i, j in the same group iff i, j have the same pattern of transitions at group level
- By Embedding
- map graph nodes $\{1, 2, \ldots, n\}$ to $\mathbb{R}^K$ then use "standard" clustering methods (e.g K-means)
- By diffusion distance

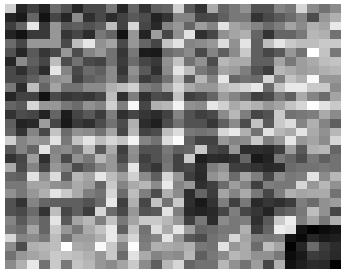
- All are equivalent (approximately) when the data is clusterable

Spectral clustering in a nutshell

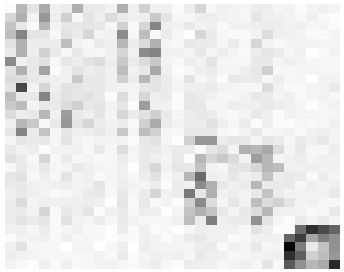


Spectral clustering in a nutshell

Similarity $\mathbf{S}$

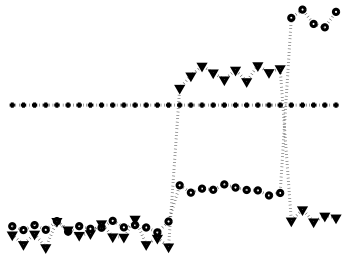


Preprocess



$P = D^{-1}S$

Top K e-vectors $\mathbf{v}^{1:K}$

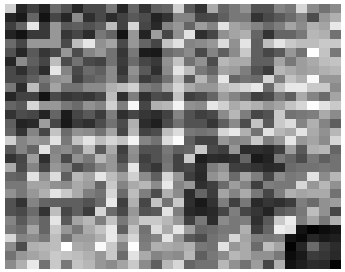


point i

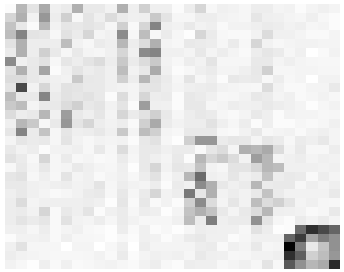
Data embedded by $\mathbf{v}$
Cluster with K-means

Spectral clustering in a nutshell

Similarity $\mathbf{S}$

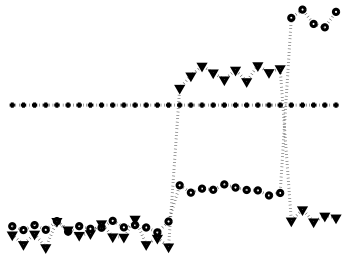


Preprocess



$P = D^{-1}S$

Top K e-vectors $\mathbf{v}^{1:K}$



point i

Data embedded by $\mathbf{v}$
Cluster with K-means

Properties of spectral clustering

- Arbitrary cluster shapes (main advantage)
- Elegant mathematically
- Practical up to medium sized problems
  - Running time (by Lanczos algorithm) $\mathcal{O}(nk)$/iteration.
- Works well when K known, not too large

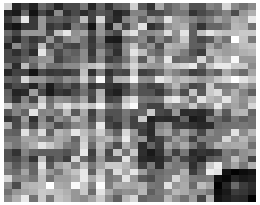
  estimating K
- Depend heavily on the similarity function (main problem)

  learning the similarities ,,,
- Outliers become separate clusters (user must adjust K accordingly!)
- Very popular, many variants which aim to improve on the above

  Diffusion maps : normalize the eigenvectors $\lambda_k^t v^k$
- Practical fix, when K large: only compute a fixed number of eigenvectors $d < K$. This avoids the effects of noise in lower ranked eigenvectors

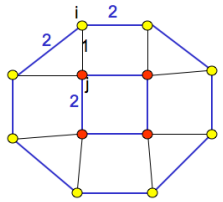
Understanding spectral clustering I

- Graph cuts Spectral clustering minimizes $MNCut(\Delta) = \sum_{k=1}^{K} \sum_{k' \neq k} \frac{Cut(C_k, C_{k'})}{D_{C_k}}$



(not the smallest K-way cut!)

- By similar behavior in the random walk on the graph



$$P_{i,red} = Pr[i \to red \,|\, i] = \sum_{j \in red} P_{ij}$$

i	$P_{i,red}$	$P_{i,yellow}$
	1/5	4/5
	1/5	4/5
	1/5	4/5
	1/5	4/5
	1/5	4/5
	1/5	4/5
	1/5	4/5
	1/5	4/5
	2/3	1/3
	2/3	1/3
	2/3	1/3
	2/3	1/3

Understanding spectral clustering II

item All are equivalent (approximately) when the data is clusterable.
Clusterability characterized by

Outline

Further issues and current trends (an incomplete list) I

- Selecting K
  - many ad-hoc methods
  - BIC (statistical model selection method) for mixture models
    theoretically unsupported in clustering
  - stability-based selection (a boostrap like method)
    - Idea: if a clustering Δ is supported by the data then it is stable to perturbations
    - theoretical results only prelimnary, but empirical evidence promising
- Non-parametric clustering
  - mixture models with unbounded K (known as Dirichlet Process Mixtures, or Bayesian Nonparametric clustering)
  - methods based on a kernel density estimator
    - find the peaks of the density Mean-Shift, Gaussian Blurring Mean-Shift
    - level-set methods (find the high density regions)
  - for similarity data **Affinity Propagation**
- Clusterability
- Algorithms with guarantees for clusterable data
- Scalable algorithms

Outline

Statistical decisions

Or learning to act in uncertainty (closer to supervised, since objective known)

- Influence diagrams
  - extend graphical models with decision nodes
- Active learning
  - Variant of supervised learning
  - Learner can choose next samples X_i to query, oracle returns the Y_i values
- Semi-supervised learning
  - Variant of supervised learning
  - In addition to labeled pairs $\mathcal{D}_L = \{(X_i, Y_i),\ i = 1 : m\}$ there are also unlabeled data $\mathcal{D}_U = \{X_j,\ j = 1 : m\}$.
  - transductive learning: infer the labels for $\mathcal{D}_U$
  - OR, use $\mathcal{D}_U \cup \mathcal{D}_L$ to learn a predictor for Y
- Reinforcement learning
  - Sequential (Markov) decisions in uncertainty (aka Markov Decision Process (MDP)

Reinforcement learning

The problem an agent learns how to act in an unknown environment

- Examples: learning to balance a beam, to drive a car, to play backgammon, navigate a maze

  Time $t = 1, 2, 3, \ldots$
  State $x_t \in \Omega$
  Action $a_t \in A$ set of available actions (may depend on x)
  Transition probability $Pr[x_{t+1}|x_t, a_t]$ (Markov transitions with T^a transition matrix, $a \in$
  Reward r_t stochastic, depends on a_t, x_t, x_{t+1}

- Goal of agent: maximize $\underbrace{E\left[\sum_{t=1}^{\infty} \gamma^t r_t\right]}_{R(x_1, a_1, a_2, \ldots)}$ where $\gamma \in (0, 1)$ is a discount factor

- Basic concepts
  - policy $\pi : \Omega \to A$ prescribes an action for every state
  - value function of policy π $V^\pi : \Omega \to \mathbb{R}$

$$V^\pi(x) \;=\; R(x, \text{follow } \pi)$$

  - Note that for each π, V_π is a linear function of expected $r(x, a)$ ($r^\pi + \gamma T^\pi V^\pi = V^\pi$)

- Classic result Bellman equation
  the optimal policy π^* satisfies

$$V^*(x), \pi*(x) \;=\; \max, \text{argmax}_{a \in A} E_{T^a}\left[r(a, x, x') + \gamma V^*(x')\right]$$

Reinforcement learning

Approaches to learning

- Finite time horizon, small Ω: dynamic programming
- Ω tractable: stochastic optimization for V^*, e.g Q-learning
- Large Ω, possibly continuous: functional approximations of V^* (may not converge!), e.g neuro-dynamic programming, deep Q-learning
- Better yet, functional approximation of the policy; i.e. let $\pi(x) = f(x, \theta)$ and optimize θ by gradient descent

A harder problem Partially observed MDP (POMDP)

- x_t not observed directly
- instead, partial information y_t
- problem becomes non-Markov, sufficient statistics are all a_t, y_t history

Connection with on-line learning, game theory, ...

What I whish I could have included

- Sparse estimation (born @UCLA)
- Model selection
- Other non-parametric models (e.g shape constrained estimation)

Thank you