ESTR4300/ ENGG5108

More on k-means Clustering

Prof. Wing C. Lau
Department of Information Engineering
wclau@ie.cuhk.edu.hk

Acknowledgements

- The slides used in this chapter are adapted from:
  - CS451 Machine Learning and Big Data, by David Kauchak, Middlebury College, Fall 2013.
  - CAS CS565 Data Mining, by Evimaria Terzi, Boston University, Fall 2013.
  - CMSC5724 Data Mining and Knowledge Discovery, by Yufei Tao, CUHK, Spring 2014.

All copyrights belong to the original author of the material.

K-means

Most well-known and popular clustering algorithm:

Start with some initial cluster centers

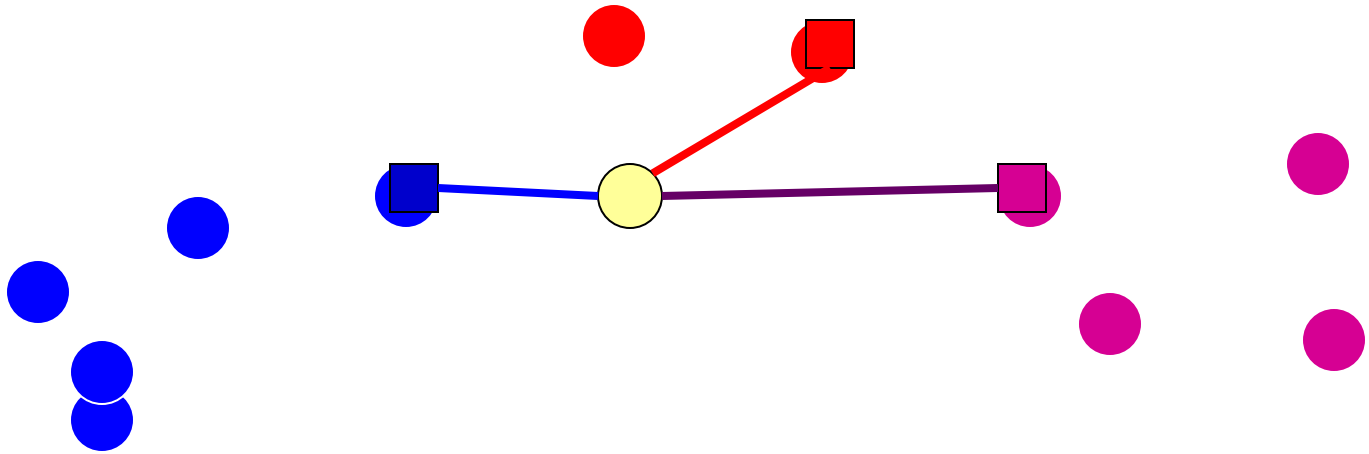
Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

K-means

Iterate:

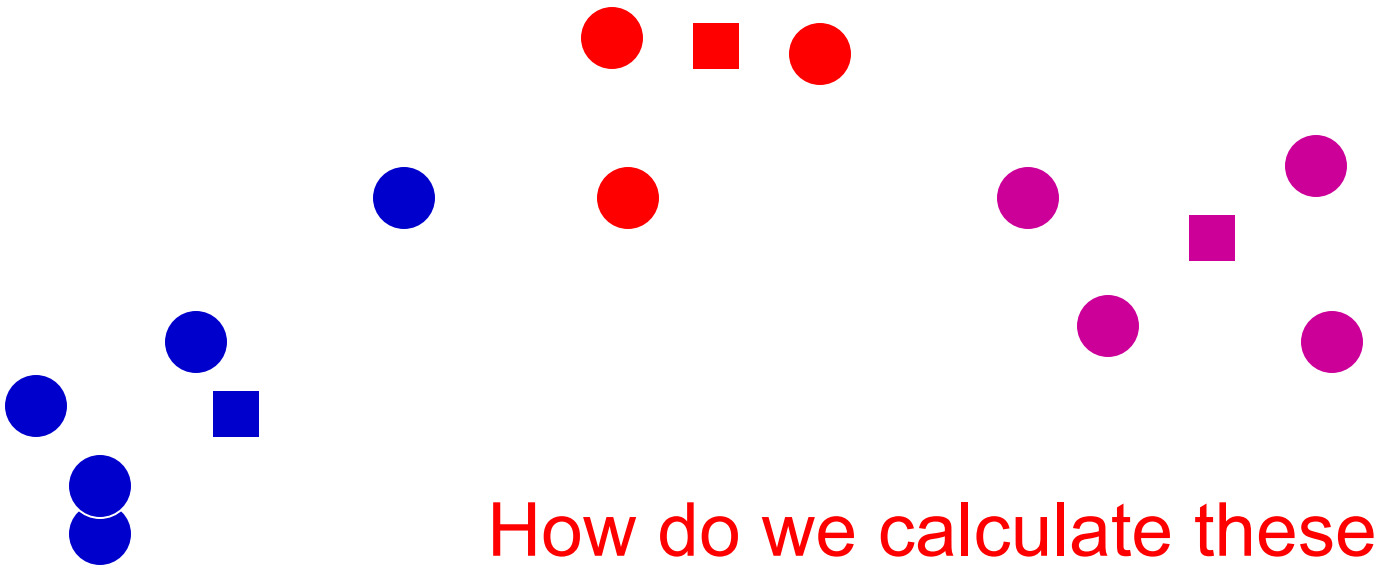
- **Assign/cluster each example to closest center**
- Recalculate centers as the mean of the points in a cluster



K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster



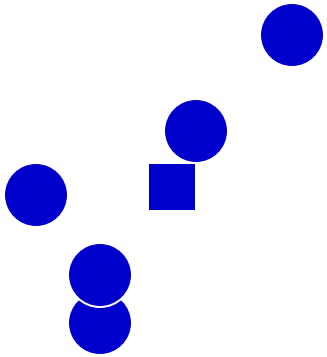
How do we calculate these?

K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

Mean of the points in the cluster:



$$\mu(\mathrm{C}) = \frac{1}{|C|}\sum_{x \in C} x$$

where:

$$x + y = \sum_{i=1}^{n} x_i + y_i \qquad \frac{x}{|C|} = \sum_{i=1}^{n} \frac{x_i}{|C|}$$

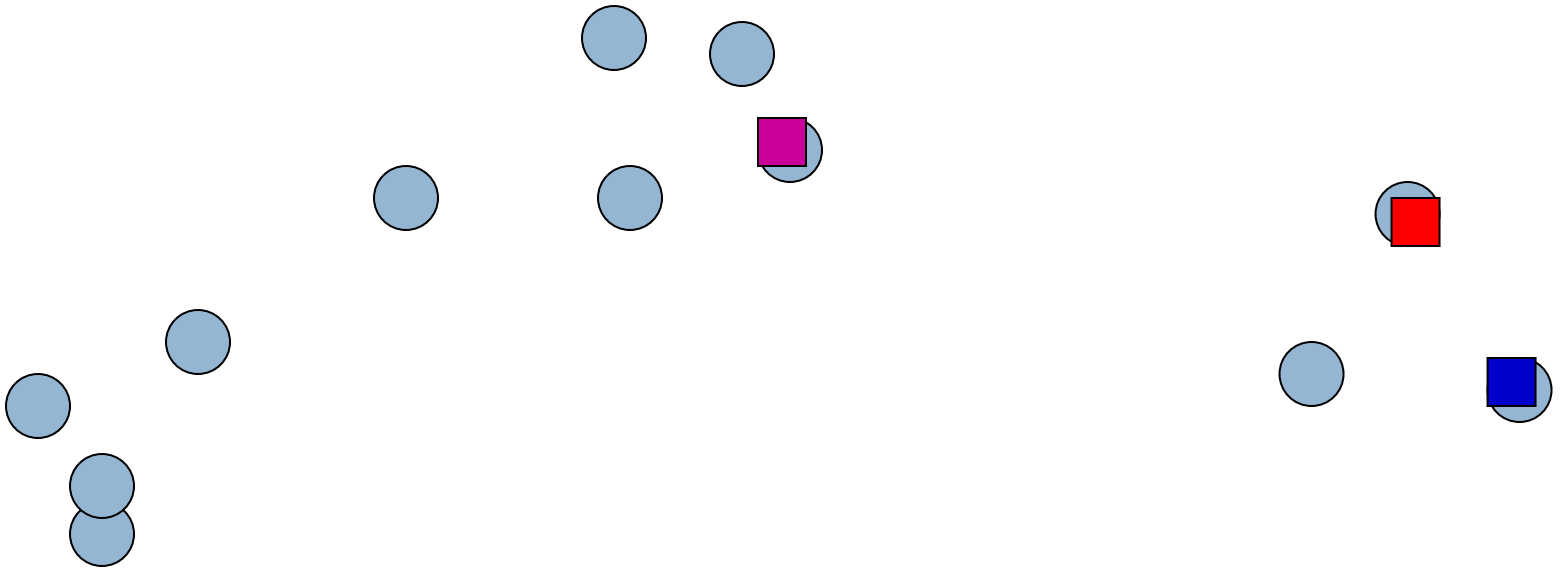
K-means loss function

K-means tries to minimize what is called the “k-means” loss function:

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

that is, the sum of the squared distances from each point to the associated cluster center

K-means: Initialize centers randomly



What would happen here?

Seed selection ideas?

Seed choice

Results can vary drastically based on random seed selection

Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings

Common heuristics

- Random centers in the space
- Randomly pick examples
- Points least similar to any existing center (furthest centers heuristic)
- **Try out multiple starting points**
- Initialize with the results of another clustering method

Furthest centers heuristic

μ_1 = pick random point

for i = 2 to K:

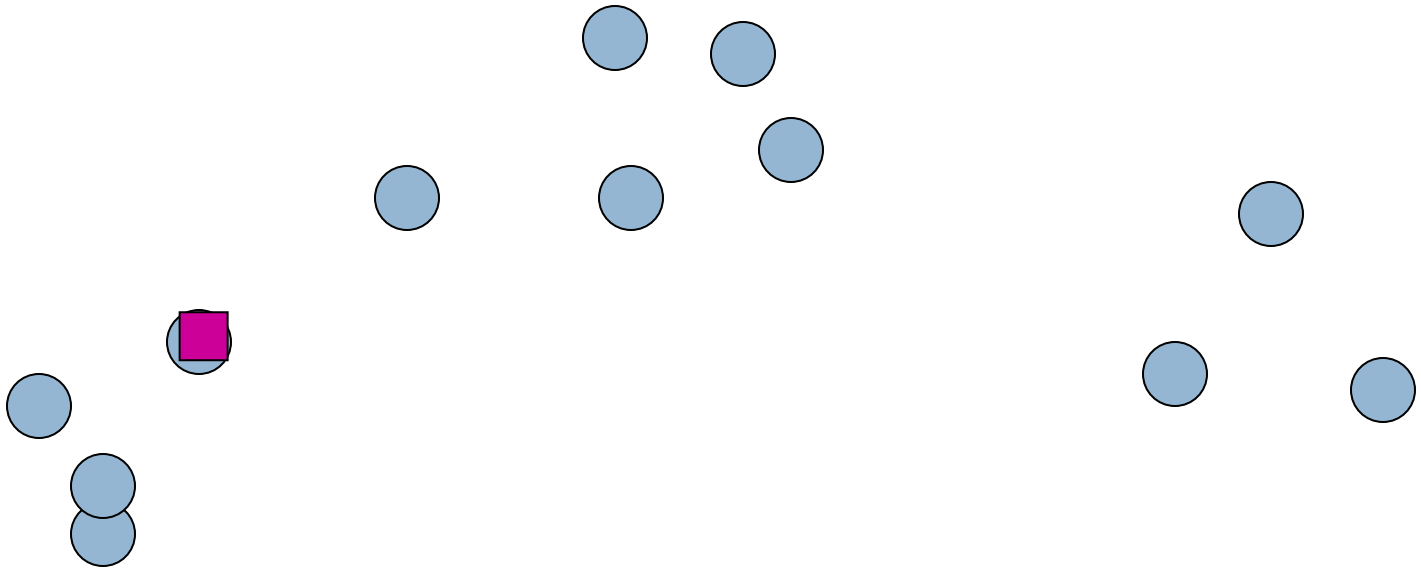
μ_i = point that is furthest from **any** previous centers

$$\mu_i = \underset{x}{\arg\max} \min_{\mu_j : 1 < j < i} d(x, \mu_j)$$

point with the largest distance to any previous center

smallest distance from x to any previous center

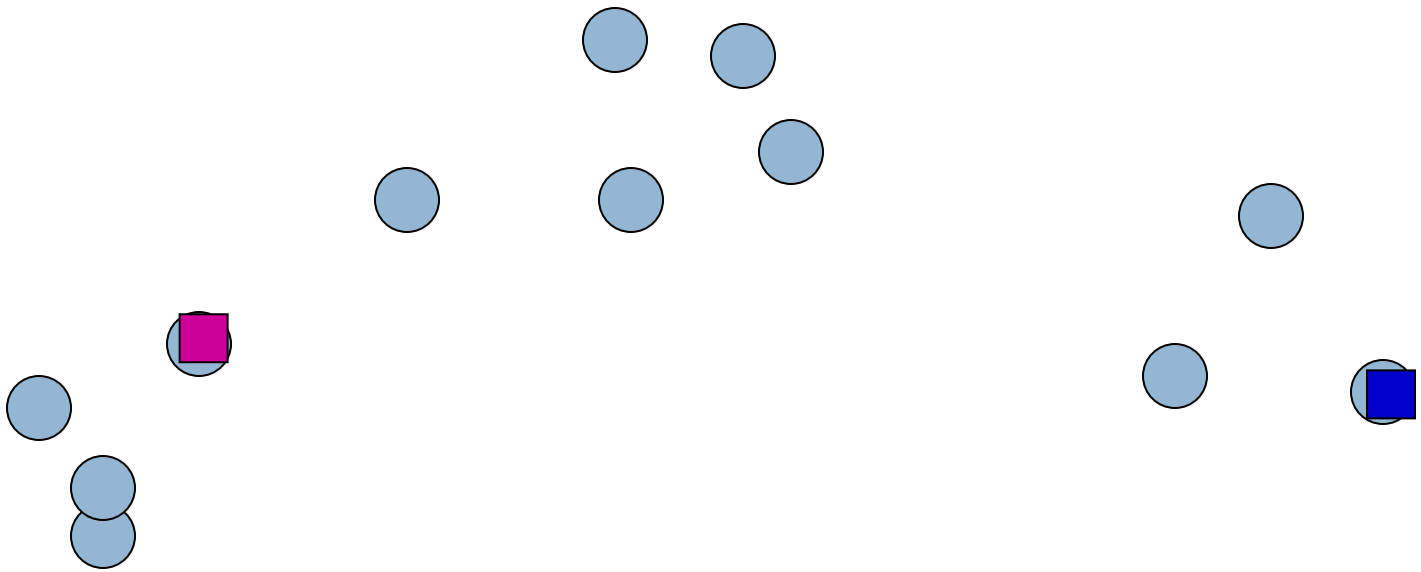
K-means: Initialize furthest from centers



Pick a random point for the first center

What point will be chosen next?

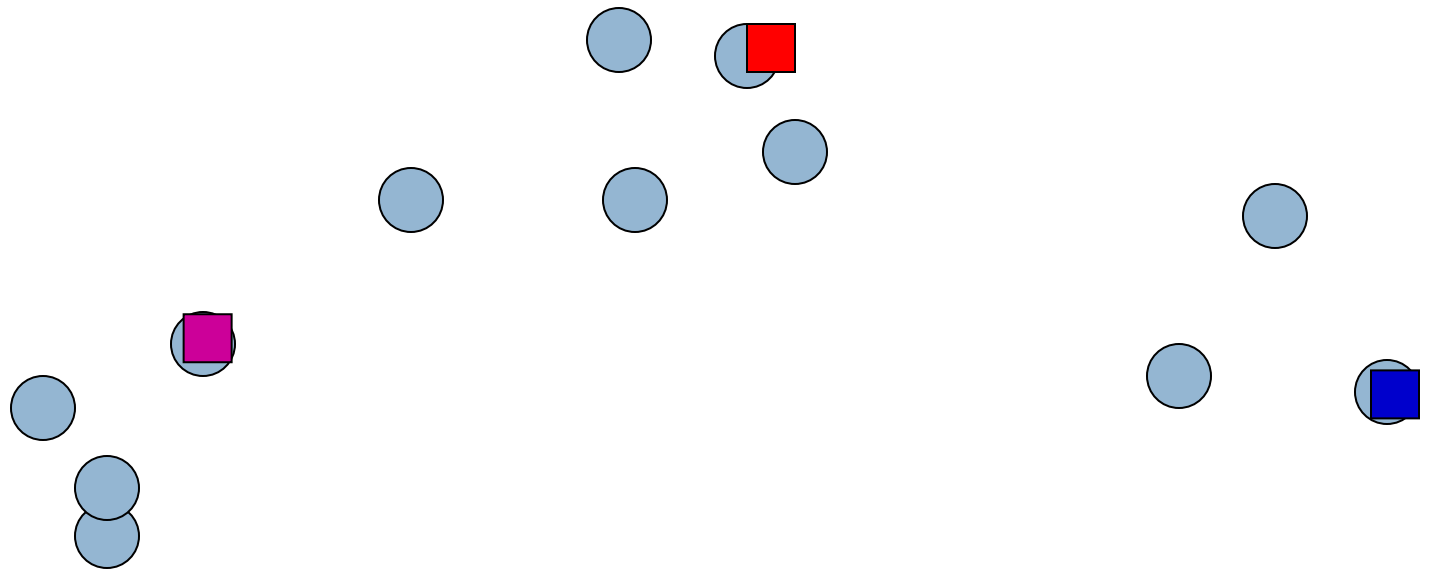
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

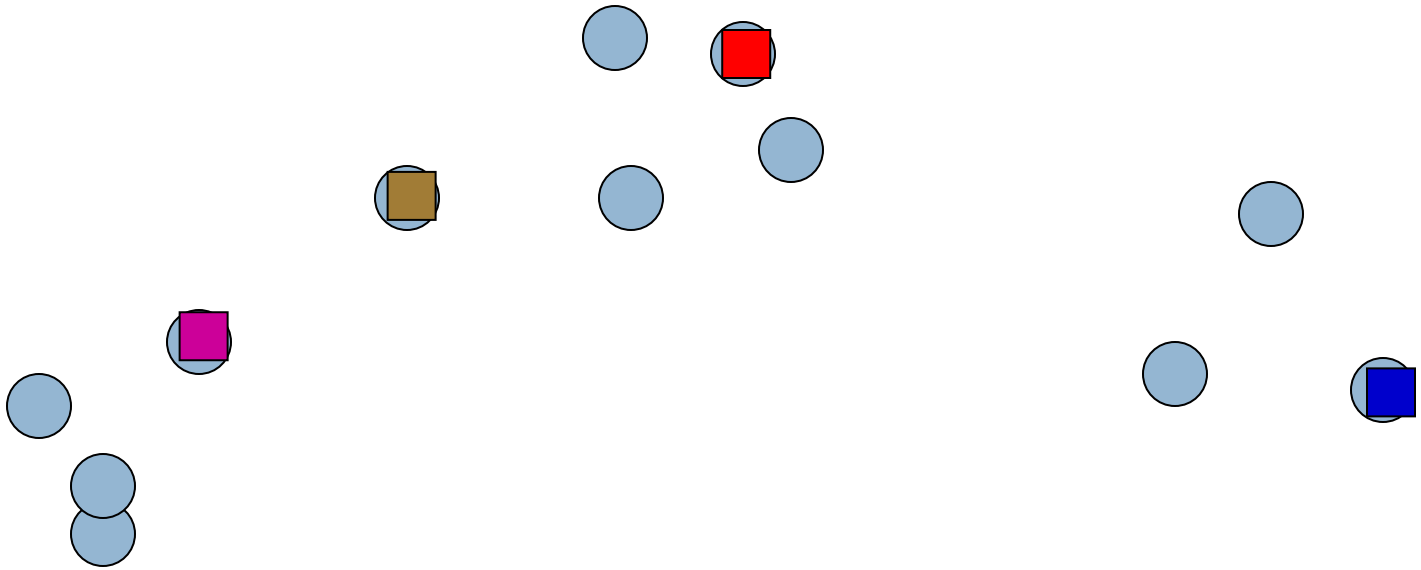
K-means: Initialize furthest from centers



Furthest point from any center

What point will be chosen next?

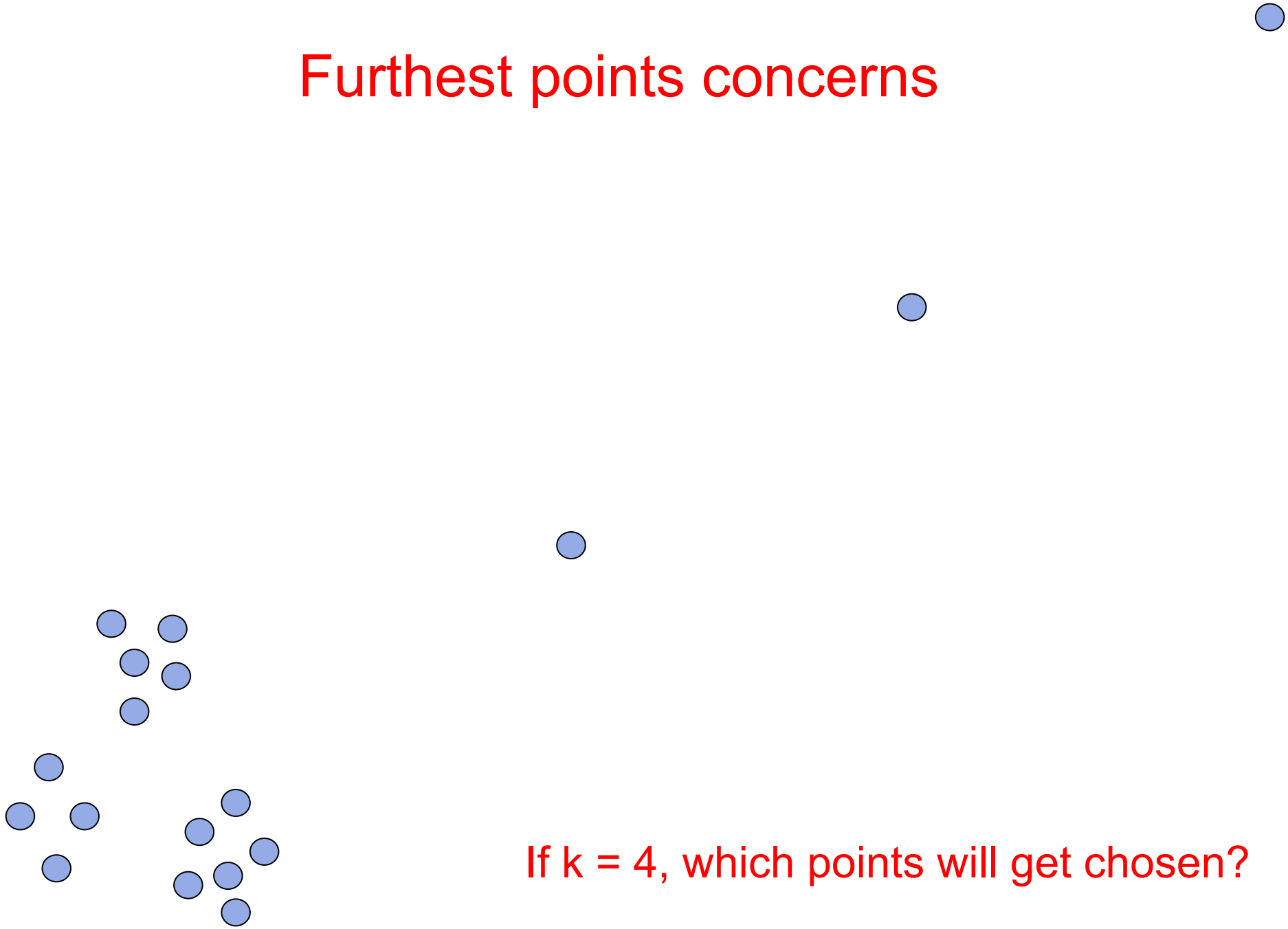
K-means: Initialize furthest from centers



Furthest point from any center

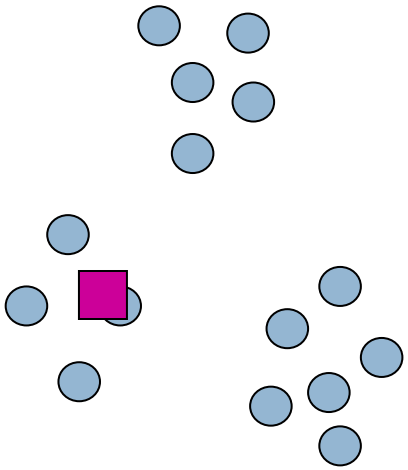
Any issues/concerns with this approach?

Furthest points concerns



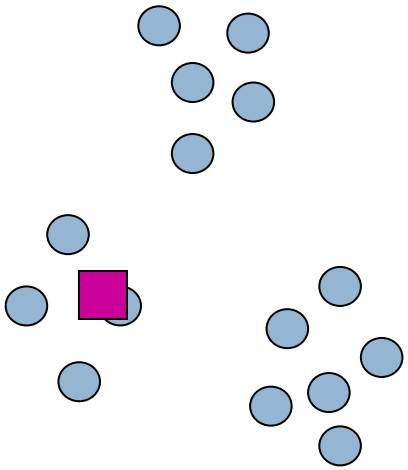
If k = 4, which points will get chosen?

Furthest points concerns



If we do a number of trials, will we get different centers?

Furthest points concerns



Doesn’t deal well with outliers

Careful Randomized Seeding: K-means++

μ_1 = pick random point

for k = 2 to **K**:

  for i = 1 to **N**:

    $D(x_i)$= min $d(x_i, \mu_{1\ldots k-1})$ // smallest distance to any center

  Set $\mu_k = x_i$ with probability ***proportional*** to $(D(x_i))^a$,

$a = 0$ random initialization

$a = \infty$ furthest-first traversal

$a = 2$ k-means++

How does this help?

K-means++

μ_1 = pick random point

for k = 2 to **K**:

for i = 1 to **N**:

$D(x_i)$= min $d(x_i, \mu_{1...k-1})$ // smallest distance to any center

Set $\mu_k = x_i$ with a probability ***proportional*** to $(D(x_i))^a$,

- With finite value of a, It becomes possible to select other points
  - if #points >> #outliers, we will pick good points
- Makes it non-deterministic, which will help with random runs
- Nice theoretical guarantees for a = 2 (aka K-mean++) !

The k-means++

David Arthur, Sergei Vassilvitskii, "k-means++: The Advantage of Careful Seeding," SODA 2007

- initialization phase:
  - choose the first center uniformly at random
  - choose next center with probability proportional to $D^2(x)$
- iteration phase:
  - iterate as in the k-means algorithm until convergence

Algorithm 1 k-means++(k) initialization.

1: $\mathcal{C} \leftarrow$ sample a point uniformly at random from X
2: **while** $|\mathcal{C}| < k$ **do**
3: Sample $x \in X$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$
4: $\mathcal{C} \leftarrow \mathcal{C} \cup \{x\}$
5: **end while**

Provable Guarantee for k-means++

Theorem:

k-means++ is O(logk) approximate in expectation

- approximation guarantee comes just from the first iteration (initialization)
- subsequent iterations can only improve cost

What's Wrong with k-Means++ ?

- Need k passes over the data (just for the initialization of k initial centers)
- In Big Data applications, both the number of data points (n) is large AND the number of categories of the data points, i.e. k is also expected to be large, e.g. 100's or 1000's !!

  => k-means++ does NOT scale and is hard to parallelize !

k-means | | [VLDB2012]

- k-means++, like other seed methods, is sequential.
  - You need to update D(p) for each unpicked p due to new point.
- Key Idea: for k-means || : many compute nodes can each handle a small set of points.
  - Each picks a few new sample points using same D(p).
- Really important and common trick: don't update after every selection; rather make O(k) selections at each round Suboptimal picks don't really matter.

  => Yield O(k log n) initial center candidates after the required number of rounds (O(log n))
  - Then Re-cluster (using k-mean++) the O(k log n) candidates to get k initial centers
  - Can be proven that log(n) rounds would suffice (to provide performance guarantee like k-mean++ (i.e. a O(log k)-factor within the optimal k-mean objective value)
    - In practice, constant number rounds, e.g. 5, is enough !

k-means | | Initialization

Algorithm 2 k-means$\|$ (k, ℓ) initialization.

1: $\mathcal{C} \leftarrow$ sample a point uniformly at random from X
2: $\psi \leftarrow \phi_X(\mathcal{C})$
3: **for** $O(\log \psi)$ times **do**
4: $\quad \mathcal{C}' \leftarrow$ sample each point $x \in X$ independently with probability $p_x = \frac{\ell \cdot d^2(x, \mathcal{C})}{\phi_X(\mathcal{C})}$
5: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$
6: **end for**
7: For $x \in \mathcal{C}$, set w_x to be the number of points in X closer to x than any other point in $\mathcal{C}$
8: Recluster the weighted points in $\mathcal{C}$ into k clusters

where

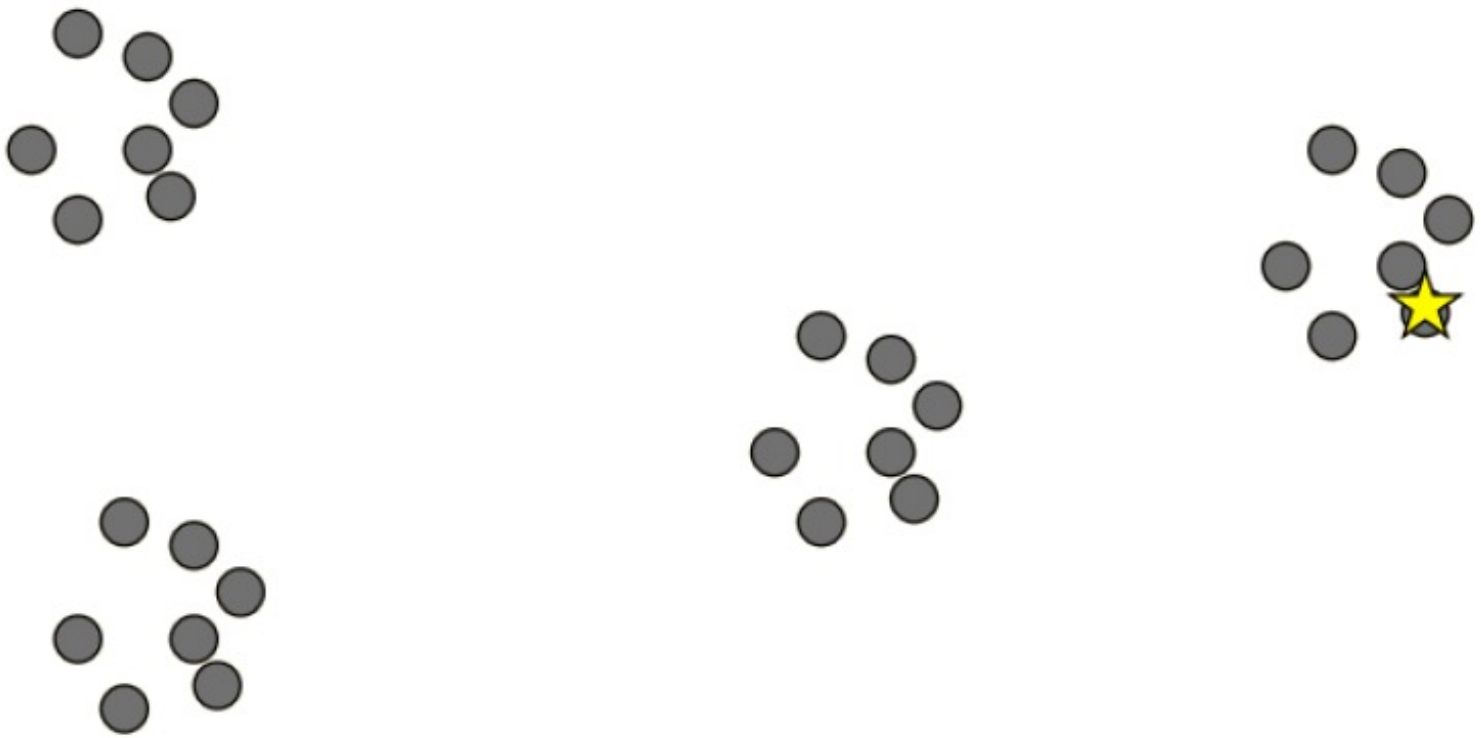
X = set of the data points to be clustered

l = oversampling factor ;

$\phi_X(C)$ = value of the k-means objective (loss) function when C is the set of cluster center

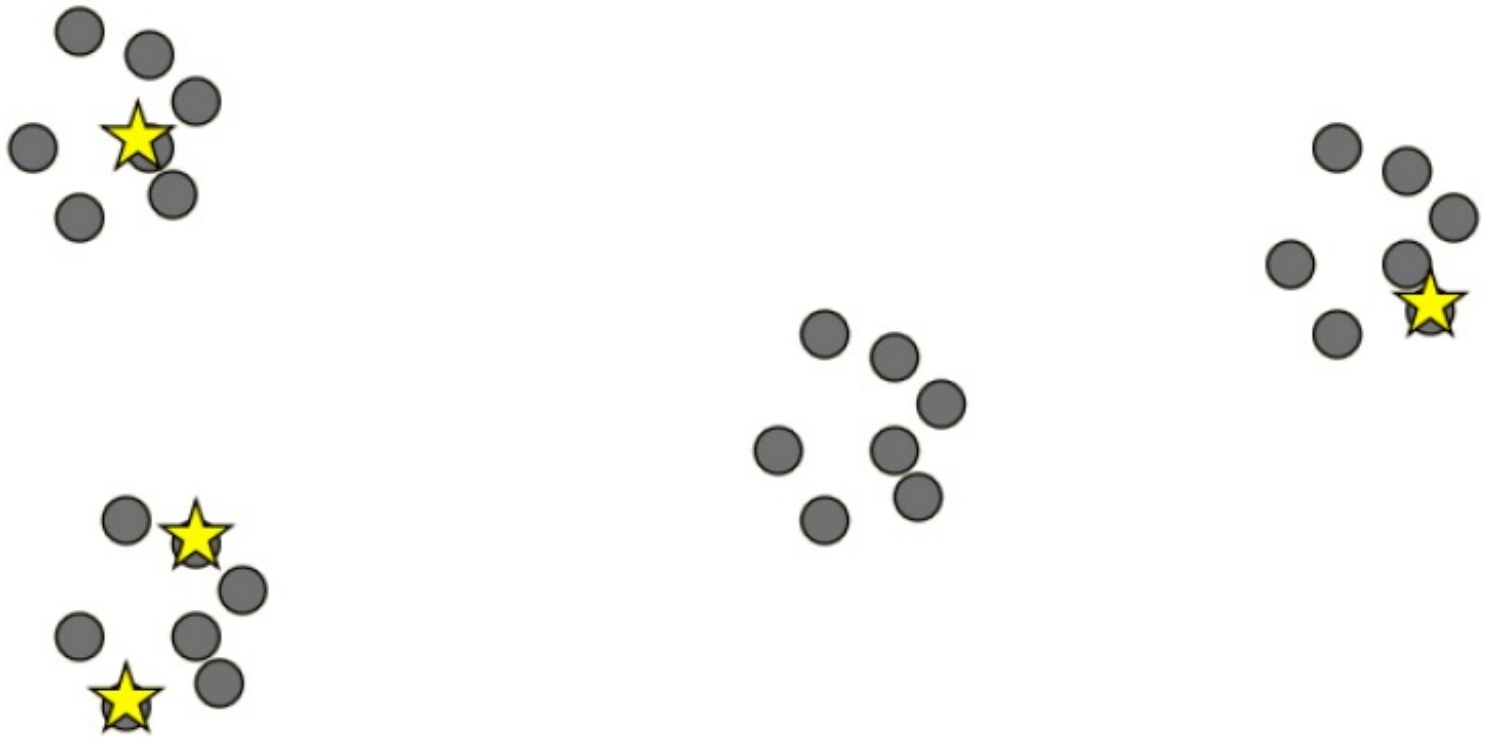
k-means++ is used for Step 8

k-means | | Initialization Example



k = 4 , Oversampling factor (*l*) = 3

k-means | | Initialization Example



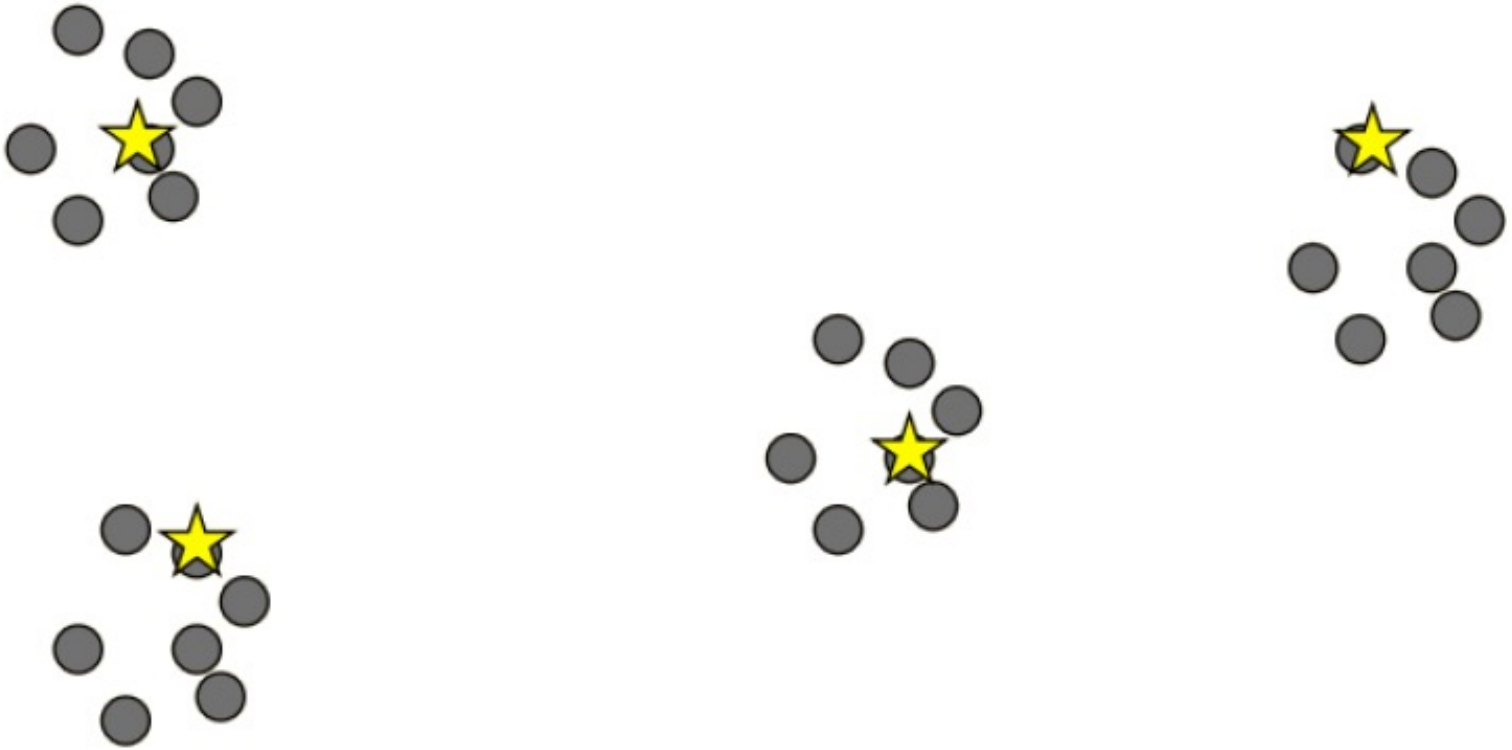
k = 4 , Oversampling factor (*l*) = 3

k-means | | Initialization Example



k = 4 , Oversampling factor (*l*) = 3

k-means | | Initialization Example



Cluster the Intermediate centers

Performance Comparison

	Clustering Cost Right After Initialization	Clustering Cost After Convergence
Random	NA	22,000
K-means++	430	65
K-means||	16	14

GAUSSMIXTURE: 10,000 points in 15 dimensions
K=50
Costs scaled down by 10^4

- K-means|| much harder than K-means++ to get confused with noisy outliers

Performance Comparison

- K-means|| reduces number of iterations even more than K-means++

	Clustering Cost Right After Initialization
Random	167
K-means++	42
K-means||	28

SPAM: 4,601 points in 58 dimensions
K=50

Performance Comparison

- K-means|| needs a small number of intermediate centers
- Better than K-means++ as soon as ~K centers chosen

	Clustering Cost (Scaled down by 10^{10})	Number of intermediate centers	Time (In Minutes)
Random	6.4×10^7	NA	489
K-means++	1.9	1.47×10^6	1022
K-means||	1.5	3604	87

KDDCUP1999: 4.8M points in 42 dimensions
K=1000

Variations of k-means

- No reason of NOT using k-means++ or k-means|| in practice !
- Also, if one recomputes the centroid after every membership assignment (rather than after all points are re-assigned), it can improve speed of convergence of *k*-means
  - More difficult to parallelize though !
- The k-median problem and the k-medoids algorithm
- The k-center problem

The k-median problem

- consider set $X=\{x_1,\ldots,x_n\}$ of n points in R^d
- assume that the number k is given
- problem:
  - find k points $c_1,\ldots,c_k$ (named medians)
  - and partition X into $\{X_1,\ldots,X_k\}$ by assigning each point x_i in X to its nearest cluster median,
  - so that the cost

$$\sum_{i=1}^{n} \min_{j} ||x_i - c_j||_2 = \sum_{j=1}^{k} \sum_{x \in X_j} ||x - c_j||_2$$

is minimized

The k-median problem

Given a collection $\{\mathbf{x}^{(i)}\}$ of N points in $\mathbb{R}^m$, their geometric median is a point $\mathbf{y}$ minimizing the sum of Euclidean distances to each point:

$$L = \sum_{i=1}^{N} \|\mathbf{x}^{(i)} - \mathbf{y}\|.$$

Each Euclidean distance in this sum is indeed a L_2 norm of a vector in $\mathbb{R}^m$. But consider a vector $\mathbf{d} \in \mathbb{R}^N$, whose coordinates are given by these distances: $d_i = \|\mathbf{x}^{(i)} - \mathbf{y}\|$. Then the same cost function L can be equivalently written as the L_1 norm of this vector:

$$L = \sum_{i=1}^{N} d_i = \sum_{i=1}^{N} |d_i| = \|\mathbf{d}\|_1.$$

That is, I believe, why the geometric median is called a L_1 estimator.

https://stats.stackexchange.com/questions/113239/why-is-the-geometric-median-called-the-l-1-estimator

The k-medoids algorithm (aka PAM = Partitioning Around Medoids)

1. randomly (or with another method) choose k medoids $\{c_1,\ldots,c_k\}$ from the original dataset X
2. assign the remaining n-k points in X to their closest medoid c_j
3. for each cluster, replace each medoid by a point in the cluster that improves the cost
4. repeat (go to step 2) until convergence

The k-center problem

- consider set $X=\{x_1,\ldots,x_n\}$ of n points in R^d
- assume that the number k is given
- problem:
  - find k points $c_1,\ldots,c_k$ (named centers)
  - and partition X into $\{X_1,\ldots,X_k\}$ by assigning each point x_i in X to its nearest cluster center,
  - so that the cost

is minimized $$\max_{i=1}^{n} \min_{j=1}^{k} ||x_i - c_j||_2$$

Properties of the k-center problem

- NP-hard for dimension $d \geq 2$
- for $d=1$ the problem is solvable in polynomial time
- a simple combinatorial algorithm works well

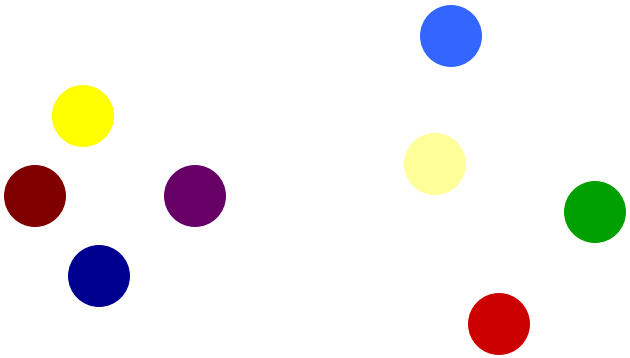
How Many Clusters?

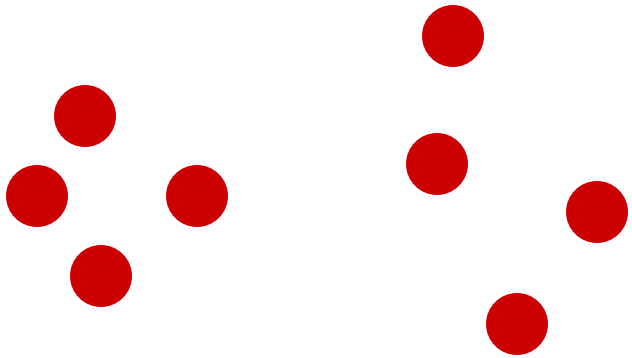
Number of clusters *K* must be provided

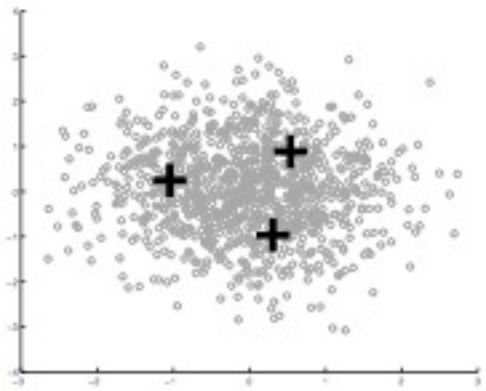
How about picking K to minimize the Loss (error) function of K-mean ?

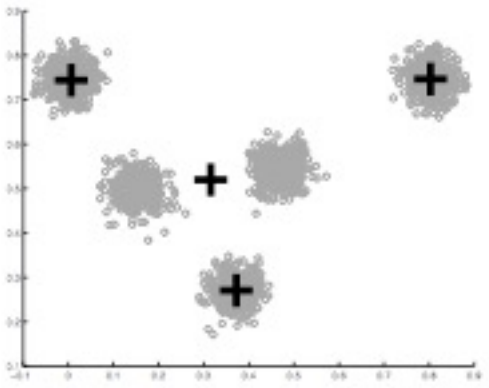
Unfortunately, the Loss function of K-means always = 0 for K = n, i.e. the model will "overfit" if assumes too-much degree of freedom, namely, K is too large !

In general, how to avoid "overfitting" when designing models?









Many approaches

Regularization!!!



Statistical test

k-means loss revisited

K-means is trying to minimize:

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

What happens when k increases?

k-means loss revisited

K-means is trying to minimize:

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

Loss goes down!

Making the model more complicated allows us more flexibility, but can "overfit" to the data

k-means loss revisited

K-means is trying to minimize:

$$loss_{kmeans} = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$



2 regularization options

$$loss_{BIC} = loss_{kmeans} + K \log N$$

$$loss_{AIC} = loss_{kmeans} + KN$$

What effect will this have?
Which will tend to produce smaller k?

k-means loss revisited

2 regularization options

$$loss_{BIC} = loss_{kmeans} + K \log N$$

$$loss_{AIC} = loss_{kmeans} + KN$$

AIC penalizes increases in K more harshly

Both require a change to the K-means algorithm

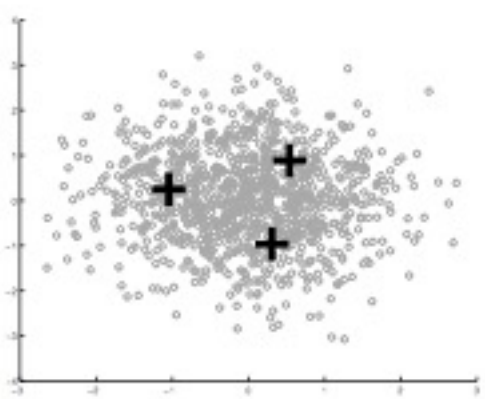
Tend to work reasonably well in practice if you don't know K

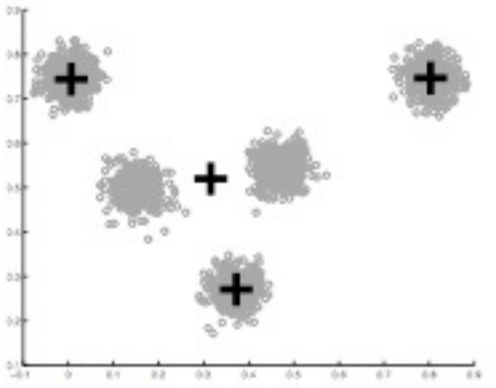
An Example of the Statistical approach: The G-means algorithm [G Hamerly, NIPS2004]

Assume data is Gaussian (i.e. spherical)

Test for this

- Testing in high dimensions doesn't work well
- Testing in lower dimensions does work well



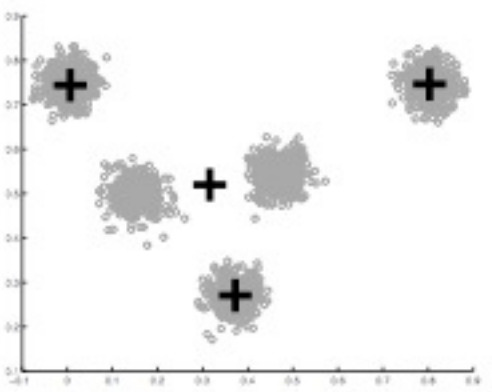


ideas?

Project to one dimension and check

For each cluster, project down to one dimension

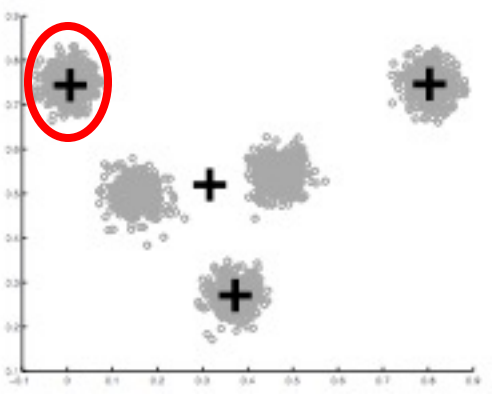
- Use a statistical test to see if the data is Gaussian



Project to one dimension and check

For each cluster, project down to one dimension

- Use a statistical test to see if the data is Gaussian

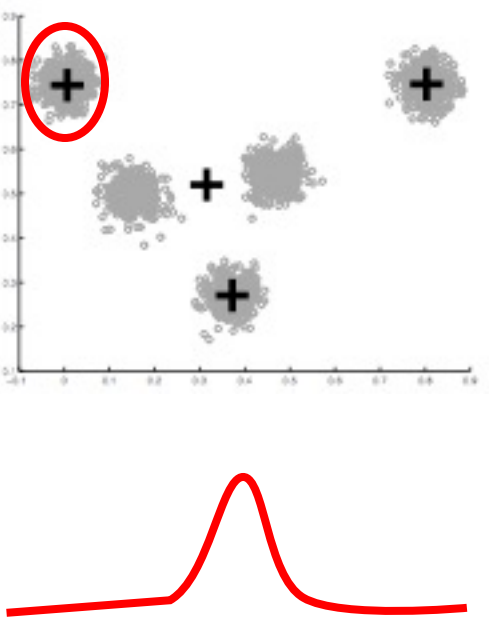


What will this look like projected to 1-D?

Project to one dimension and check

For each cluster, project down to one dimension

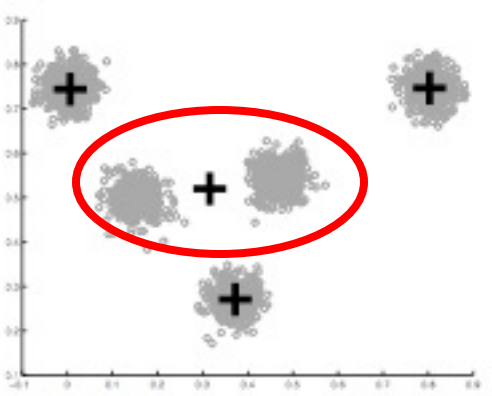
- Use a statistical test to see if the data is Gaussian



Project to one dimension and check

For each cluster, project down to one dimension

- Use a statistical test to see if the data is Gaussian

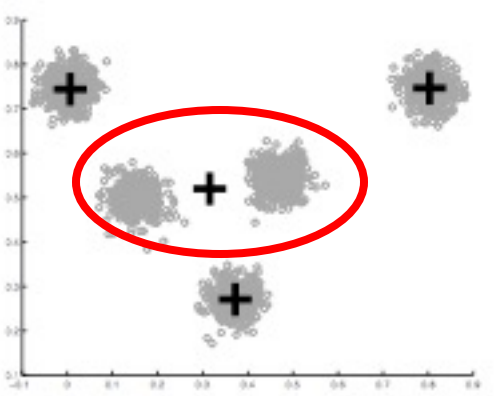


What will this look like projected to 1-D?

Project to one dimension and check

For each cluster, project down to one dimension

- Use a statistical test to see if the data is Gaussian

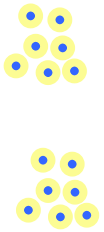




Project to one dimension and check

For each cluster, project down to one dimension

- Use a statistical test to see if the data is Gaussian

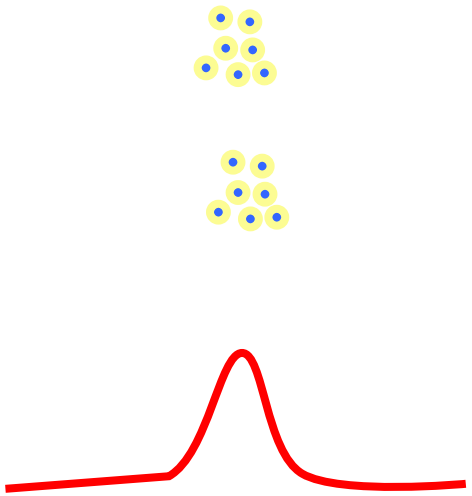


What will this look like projected to 1-D?

Project to one dimension and check

For each cluster, project down to one dimension

- Use a statistical test to see if the data is Gaussian

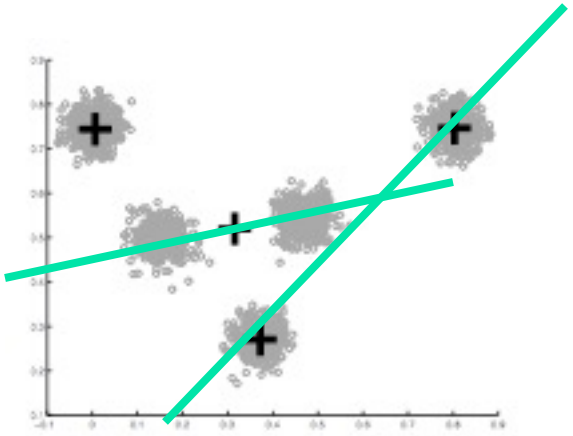


Solution?

Project to one dimension and check

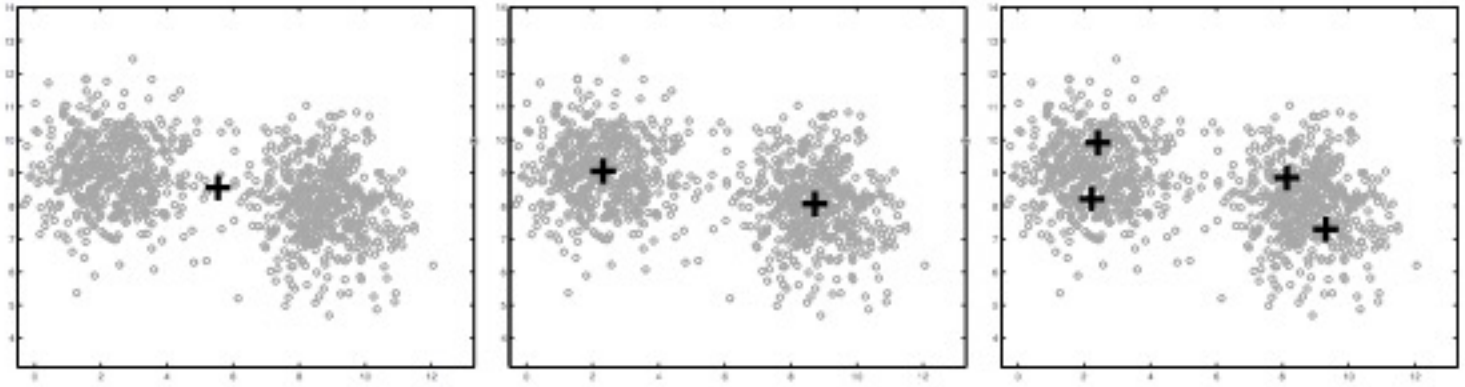
For each cluster, project down to one dimension

- Use a statistical test to see if the data is Gaussian



Chose the dimension of the projection as the dimension with highest variance

G-means on synthetic data



Compared to other approaches

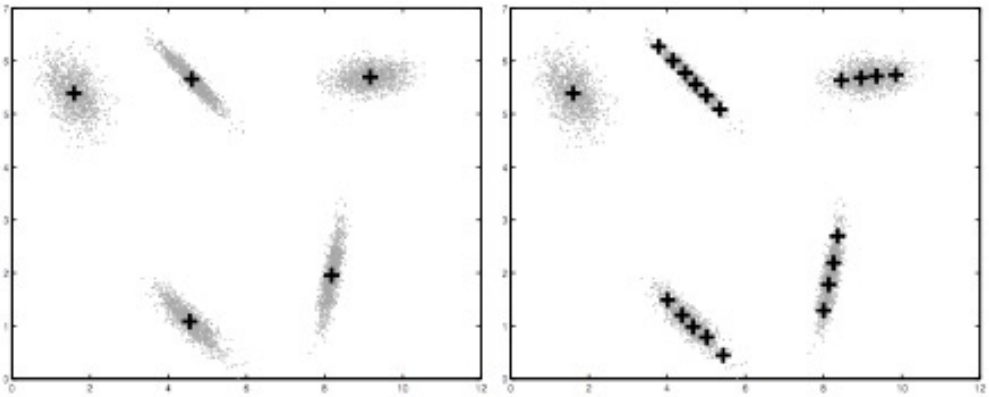
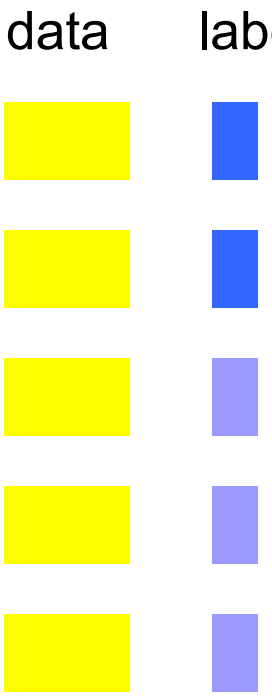


Figure 4: 2-d synthetic dataset with 5 true clusters. On the left, G-means correctly chooses 5 centers and deals well with non-spherical data. On the right, the BIC causes X-means to overfit the data, choosing 20 unevenly distributed clusters.

Common approach: use labeled data

Use data with known classes

- For example, document classification data



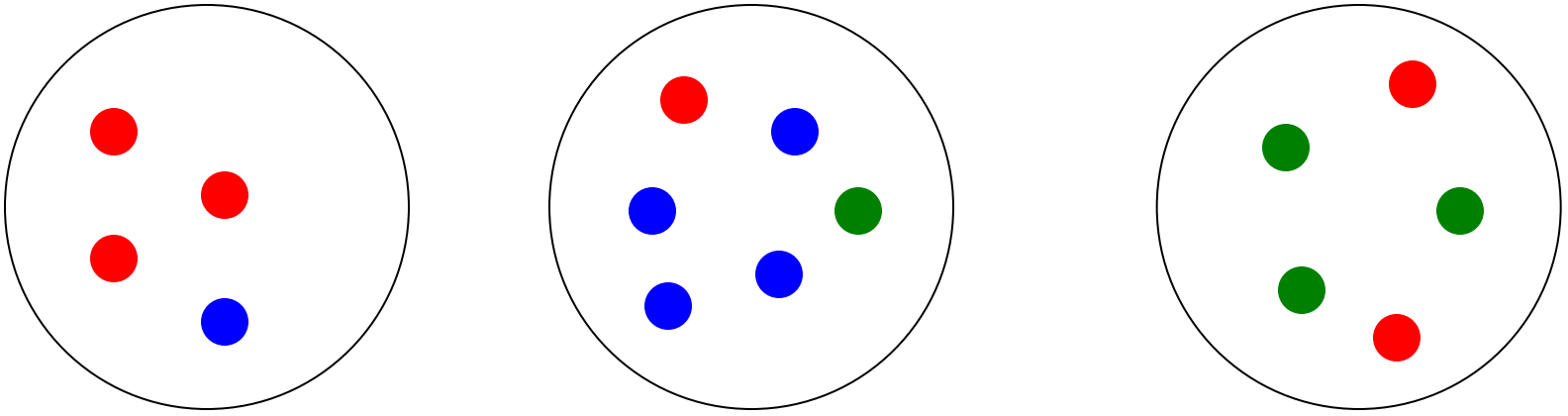
If we clustered this data (ignoring labels) what would we like to see?

Reproduces class partitions

How can we quantify this?

Common approach: use labeled data

Purity, the proportion of the dominant class in the cluster



Cluster I: Purity = 1/4 (max(3, 1, 0)) = 3/4

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Overall purity?

Overall purity

Cluster I: Purity = 1/4 (max(3, 1, 0)) = 3/4

Cluster II: Purity = 1/6 (max(1, 4, 1)) = 4/6

Cluster III: Purity = 1/5 (max(2, 0, 3)) = 3/5

Cluster average:

$$\frac{\frac{3}{4}+\frac{4}{6}+\frac{3}{5}}{3}=0.672$$

Weighted average:

$$\frac{4*\frac{3}{4}+6*\frac{4}{6}+5*\frac{3}{5}}{15}=\frac{3+4+3}{15}=0.667$$

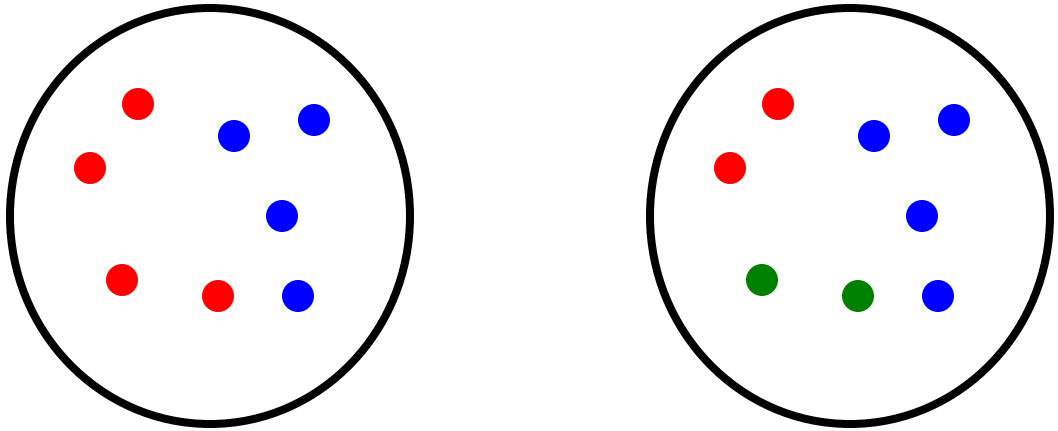
Purity issues…

Purity, the proportion of the dominant class in the cluster

Good for comparing two algorithms, but not understanding how well a single algorithm is doing, why?

- Increasing the number of clusters increases purity

Purity isn't perfect



Which is better based on purity?

Which do you think is better?

Ideas?

Common approach: use labeled data

Average entropy of classes in clusters

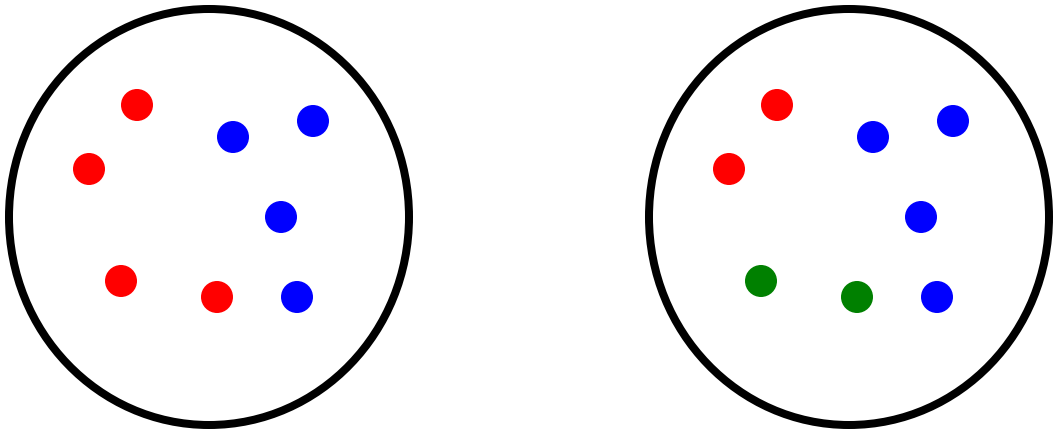
$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$

where $p(class_i)$ is proportion of class *i* in cluster

Common approach: use labeled data

Average entropy of classes in clusters

$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$

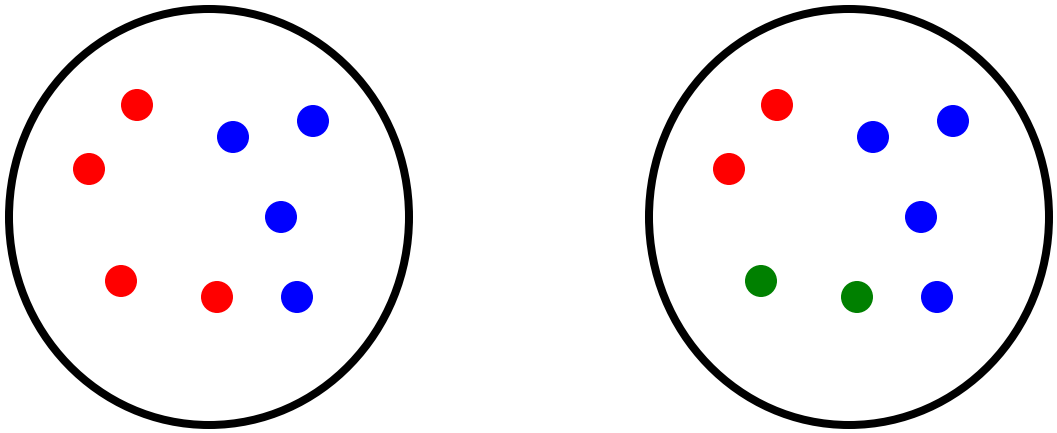


entropy?

Common approach: use labeled data

Average entropy of classes in clusters

$$entropy(cluster) = -\sum_i p(class_i)\log p(class_i)$$

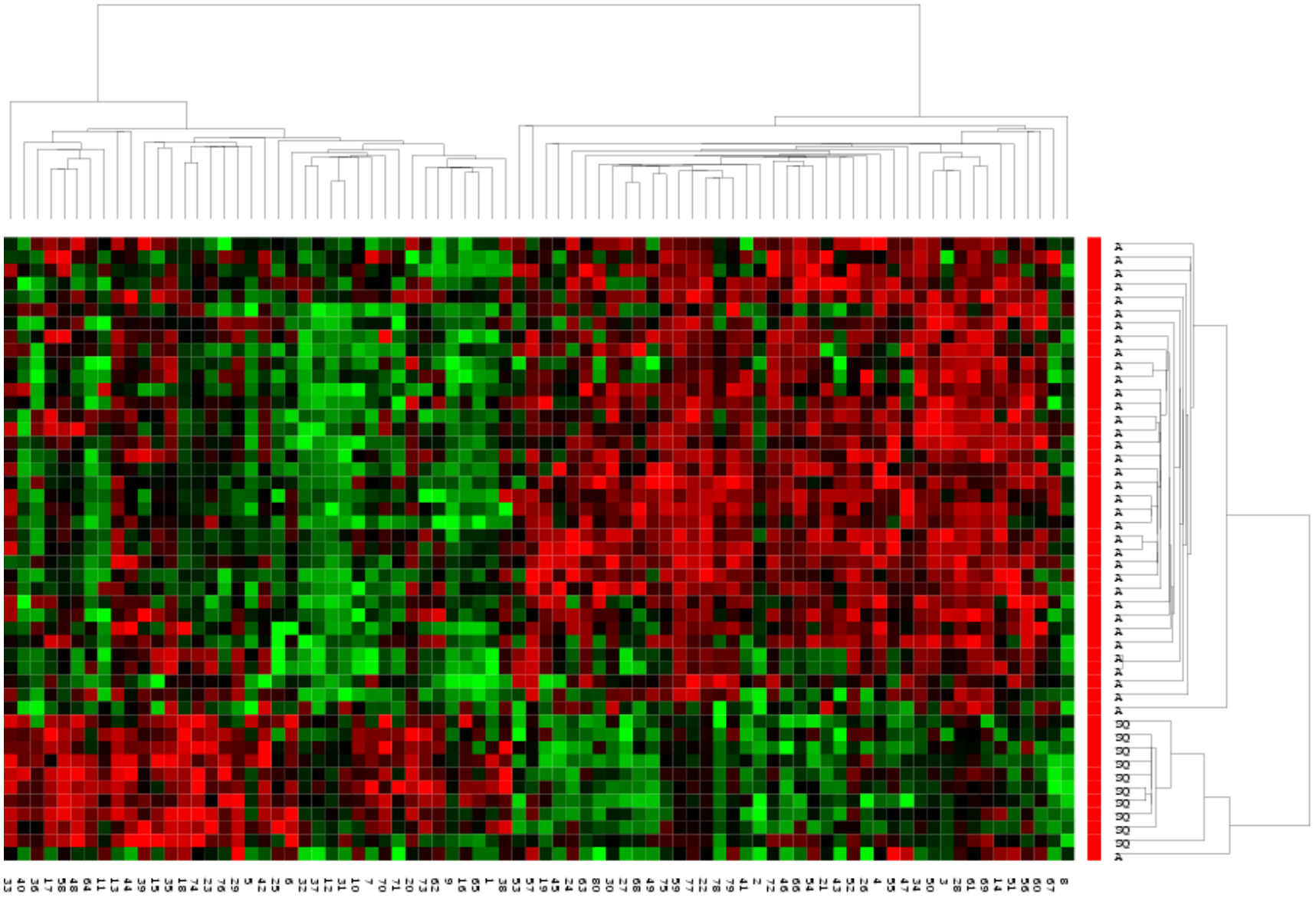


$-0.5\log 0.5 - 0.5\log 0.5 = 1$

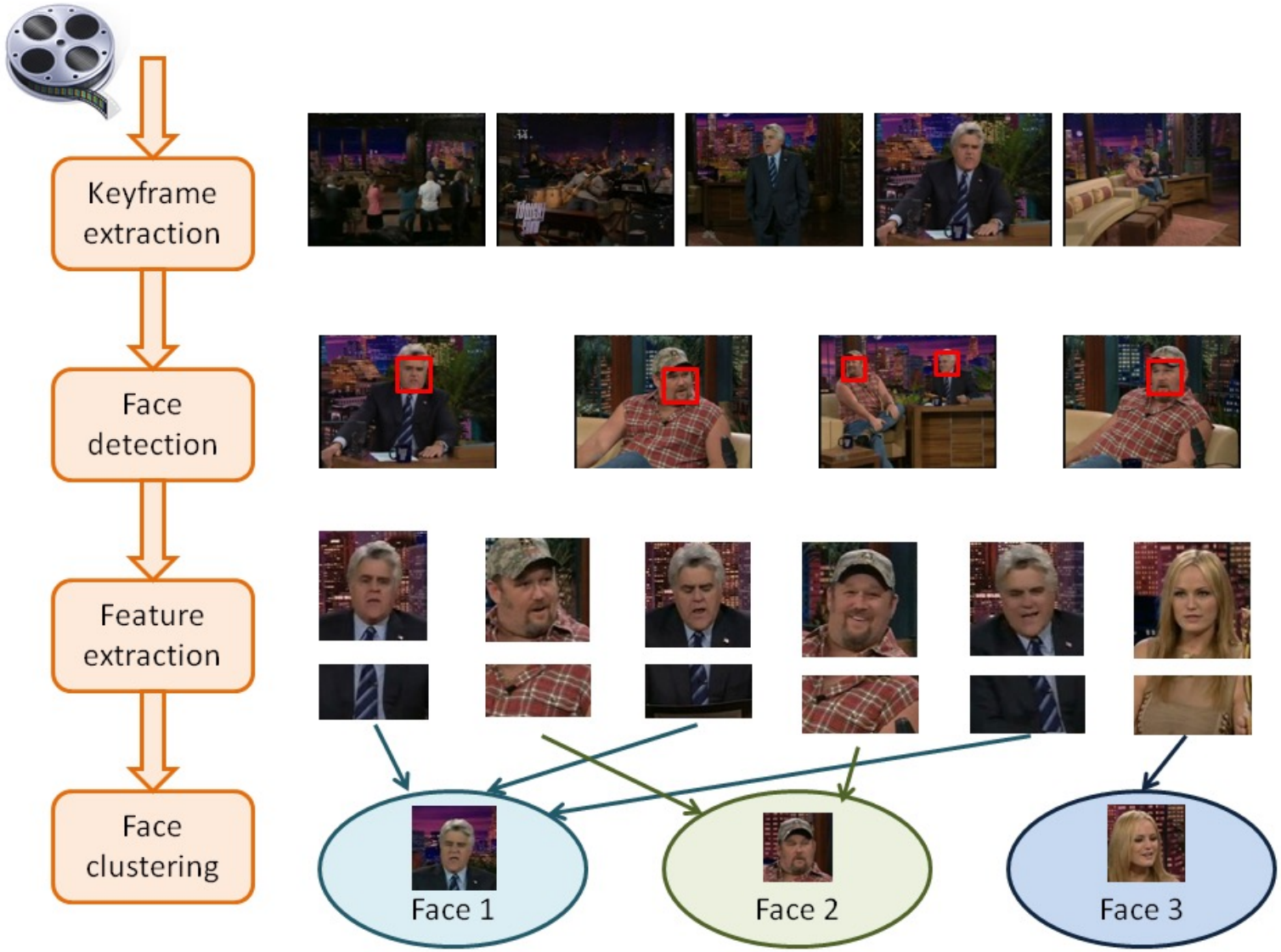
$-0.5\log 0.5 - 0.25\log 0.25 - 0.25\log 0.25 = 1.5$

Applications of Clustering

Gene expression data



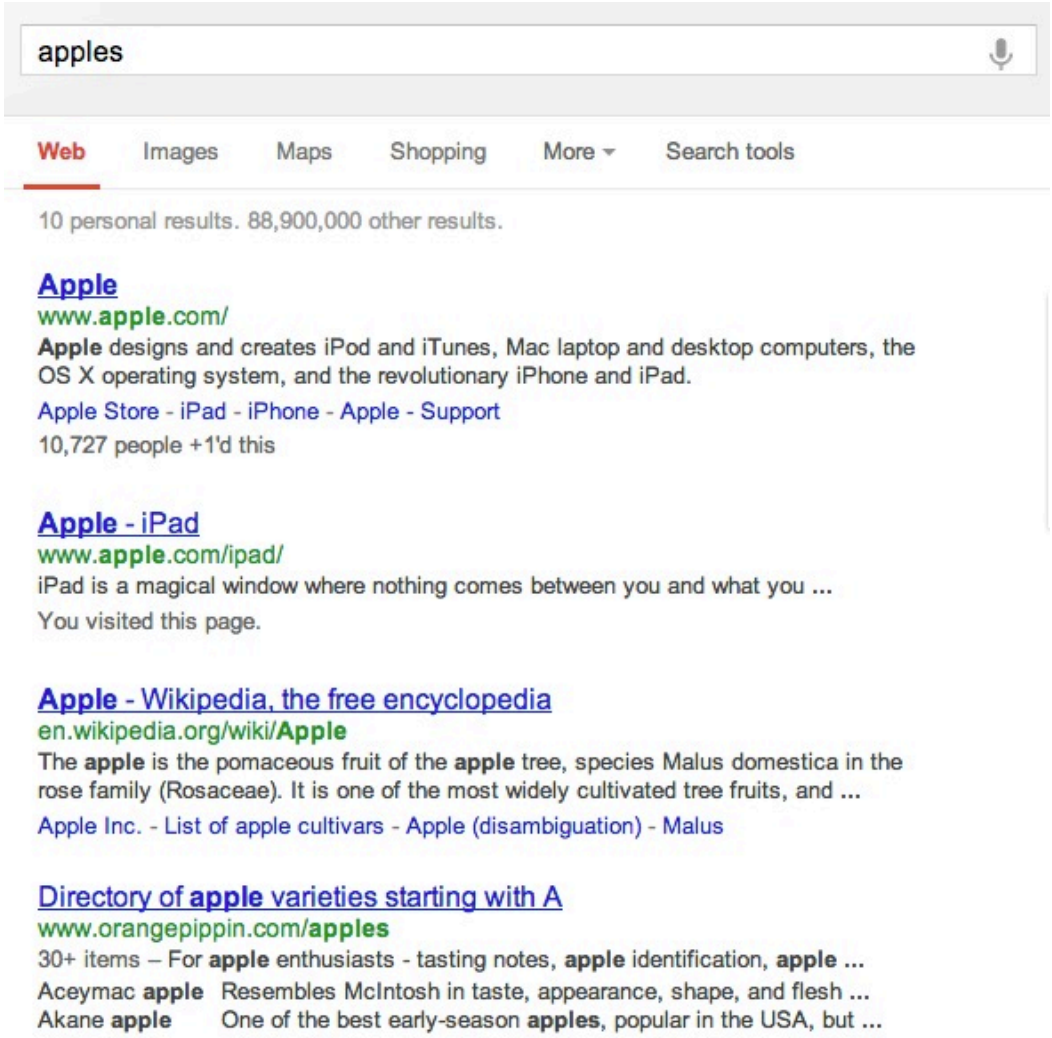
Face Clustering



Face clustering



Search result clustering



Google News

Google

News

Top Stories

- Iran
- Xbox One
- Tarun Tejpal
- Manny Pacquiáo
- Ukraine
- Kabul
- New England Patriots
- Latvia
- Derrick Rose
- Doctor Who

+ Xbox One



E! Online

See realtime coverage

Console Wars 2013: Microsoft's Xbox One vs. Sony's PlayStation 4

E! Online - 1 hour ago

The future is now! Last week, Sony released its next generation console, PlayStation 4. This weekend, Microsoft drops the much touted all-in-one media device, Xbox One. We've been geeking out over the two new systems, and compiling a report on the new ...

Xbox One sales exceed one million in first 24 hours Joystiq - by David Hinkle

Xbox One vs. PS4: A Guide to Making the Toughest Gaming Decision in Years ABC News - by Joanna Stern

Related Microsoft »



Wall Street Journal



YouTube



YouTube



Washingto...



RedOrbit



Guardian ...



The Guardian

Xbox One and Microsoft websites marred by problems on launch day

The Guardian
9 hours ago



Written by Jemima Kiss

Microsoft's Xbox One launch was marred by problems with its online services early on Friday which took down the official website Xbox.

Consumers line up for Xbox One

USA TODAY - Nov 23, 2013

Eager video game players lined up at stores across the country awaiting the arrival of Microsoft's Xbox One, a week to the day after rival Sony introduced its PlayStation 4. The console, available for sale tonight at 12:01 a.m.



Product Re...

Here are all the Xbox One voice commands

Polygon
9 hours ago



Written by Megan Farokhmanesh

Microsoft posted a guide to Xbox One voice commands, including how to navigate menus, control volume and multitask, on its Tumblr.