

PDD

Lecture 4: Clustering and MLlib

Prepared by Jacek Sroka

The plan

- What is clustering
- Similarity measures
- Clustering algorithm types
- K-means clustering
- Distributing k-means clustering to work on big data
- K-means++
- K-means||

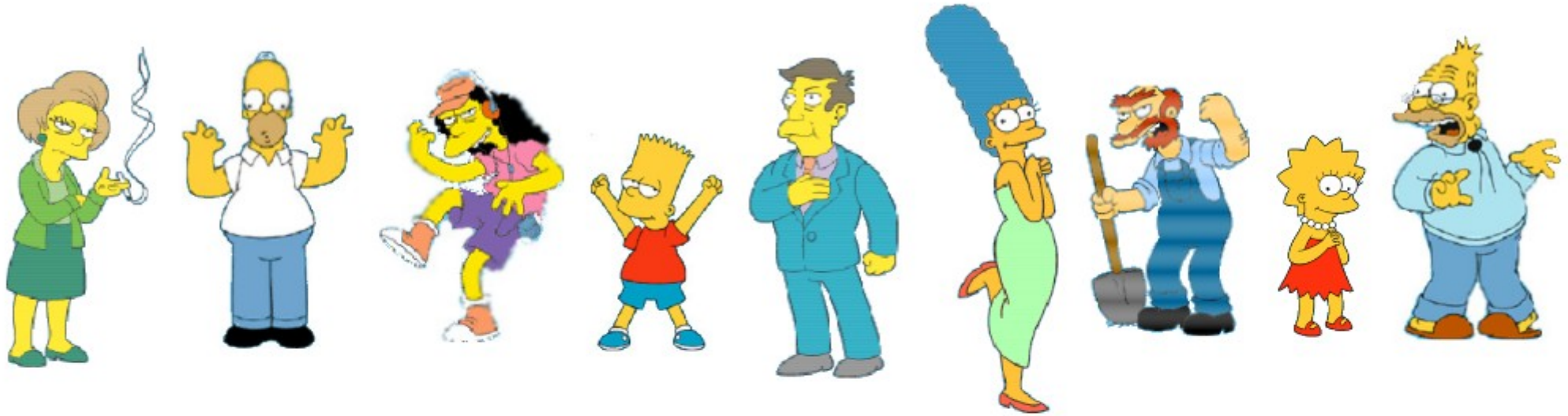
What is clustering

- Informally: finding natural groupings among objects
- Organizing data so that we have:
 - high intra-cluster similarity
 - low inter-cluster similarity
- Applications
 - Customer Segmentation
 - Look at specific characteristics of people and involve them in campaigns that have been successful with other similar people. Clustering algorithms are able to group together people with similar traits and likelihood to purchase.
 - Document Classification
 - The initial processing of the documents is needed to represent each document as a vector and uses term frequency to identify commonly used terms that help classify the document.
 - Image segmentation
 - Partitioning a digital image into multiple distinct regions containing each pixel (sets of pixels, also known as superpixels) with similar attributes.
- Unsupervised
 - Input data is not labeled (if labeled becomes classification)
 - Clustering can simplify large datasets or can segment them into smaller sets

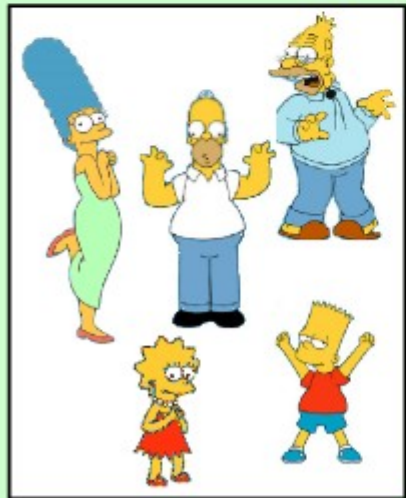
What is natural grouping among objects?

- For example let us group students, by:
 - age/studies year
 - sex
 - height
 - hair length/color
 - grades average
 - classes they attend/attended
 - ...
- Clustering is subjective
- Many ways to define similarity

What is a natural grouping among these objects?



Clustering is subjective



Simpson's Family



School Employees



Females



Males

Many ways to define similarity



Similarity

- Requirements

- $d(x,y) \geq 0$
- $d(x,y) = 0$ if and only if $x = y$
- symmetry: $d(x,y) = d(y,x)$
- triangle inequality: $d(x,z) \leq d(x,y) + d(y,z)$

- Examples

- Minkowsky distance

$$\text{1-norm distance} = \sum_{i=1}^n |x_i - y_i|$$

$$\text{2-norm distance} = \left(\sum_{i=1}^n |x_i - y_i|^2 \right)^{1/2}$$

$$\text{p-norm distance} = \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p}$$

$$\begin{aligned} \infty\text{-norm distance} &= \lim_{p \rightarrow \infty} \left(\sum_{i=1}^n |x_i - y_i|^p \right)^{1/p} = \\ &= \max(|x_1 - y_1|, |x_2 - y_2|, \dots, |x_n - y_n|) \end{aligned}$$

Distance examples

- Cosine distance/similarity

(Cosine similarity can be seen as a method of normalizing document lengths during comparison)

$$\mathbf{A} \cdot \mathbf{B} = \|\mathbf{A}\| \|\mathbf{B}\| \cos(\theta)$$

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

$$\text{distance} = 1 - \text{similarity}$$

Note: If we normalize all vectors to unit length, then: $\cos(\theta) = 1 - \frac{d(\mathbf{A}, \mathbf{B})^2}{2}$ where d is 2-norm.
(see <https://stats.stackexchange.com/questions/299013/cosine-distance-as-similarity-measure-in-kmeans>)

- Jaccard distance

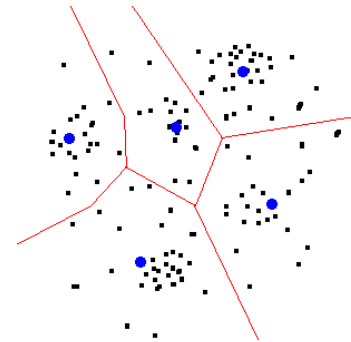
$$d_J(A, B) = 1 - J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

- <https://cran.r-project.org/web/packages/philentropy/index.html>

Clustering methods

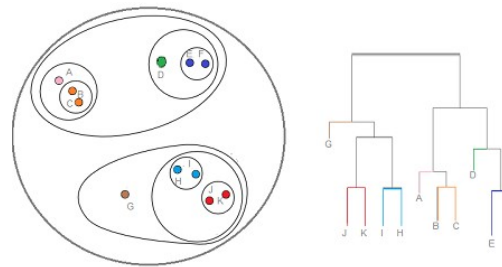
- Centroid-based Clustering

- K-means
- Fuzzy clustering
- K-medoids



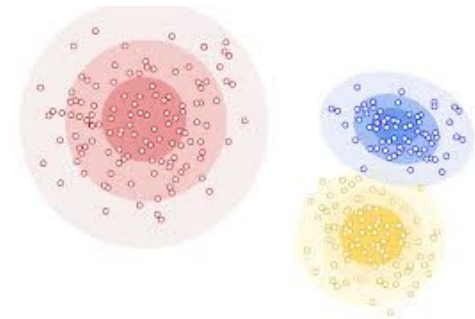
- Hierarchical clustering

- Bottom-Up (agglomerative)
- Top-Down (divisive)
- Different measures of distance of point to cluster
- Need to choose the level with right number of clusters



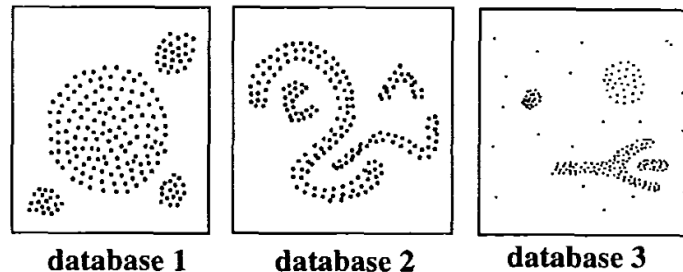
- Distribution-based Clustering

- Assumes data is composed of known distributions, such as Gaussian distributions



- Density-based Clustering

- DBSCAN



- Bi-clustering, co-clustering, bi-dimensional clustering, two-way clustering or subspace clustering

K-means clustering

- One of the top 10 algorithms in data mining

$$X = \{x_1, \dots, x_n\} \text{ where } x_i \in \mathbb{R}^d$$

$d(x_i, x_j)$ distance between points

$d(x, Y) = \min_{y \in Y} d(x, y)$ distance of a point from a set


$$\text{centroid}(Y) = \frac{1}{|Y|} \sum_{y \in Y} y$$

Not always exists/defined
Needs to be consistent with
distance (k-medoids)

for $Y, C \subseteq X$ we define $\Phi_Y(C) = \sum_{y \in Y} d^2(y, C)$

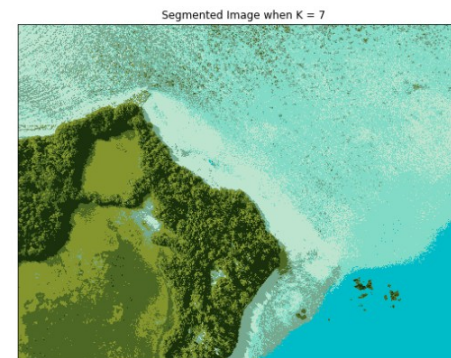
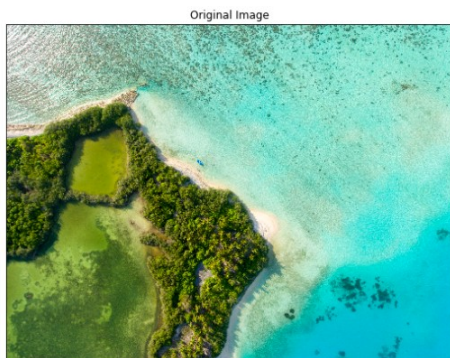
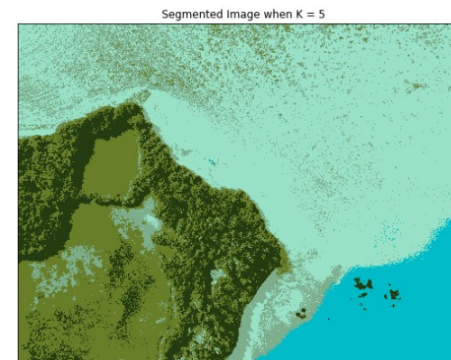
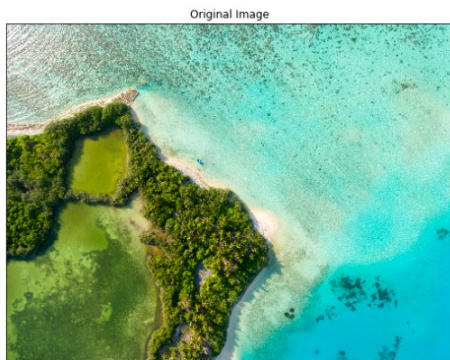
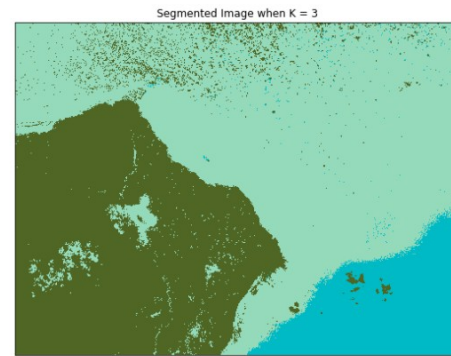
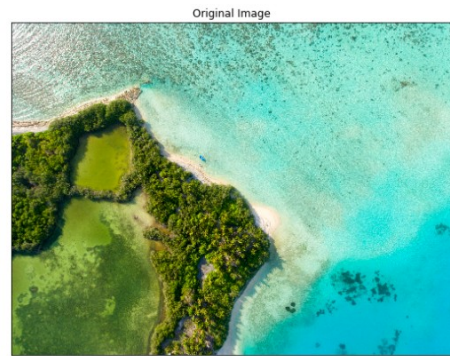
Goal: find $\{C = c_1, \dots, c_k\}$ such that $\Phi_X(C)$ is minimized

Within-Cluster-Sum of Squared Distances



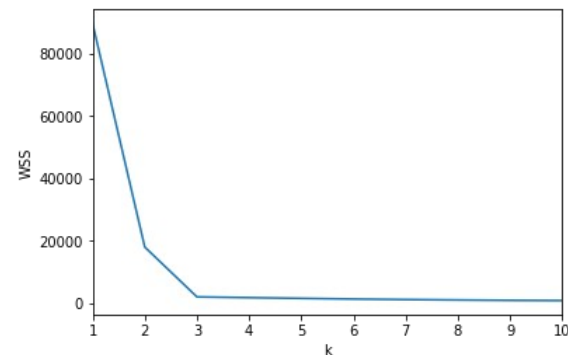
K-means segmentation example

- See: Introduction to Image Segmentation with K-Means clustering, Nagesh Singh Chauhan



K-means

- Centers define clusters
 - assign points to closest center
- Problem is NP-hard
- We will look for α -approximations $\Phi_Y(C) \leq \alpha \Phi^*$
- Let us start with Lloyd's iteration:
 - Pick k centers at random
 - Assign points to centers
 - Update centers to centroids
 - Repeat until centers do not change much
- Advantages
 - Simple, easy to compute
- Problems
 - Need to know k in advance
 - Use Within-Cluster-Sum of Squared Distance
 - Or silhouette method
 - Can get stuck on local minimum
 - Requires **data normalization**
 - Partitions the space, works worser for non-spherical data



Silhouette method

For each point i we compute its silhouette value which is between +1 and -1 (how well it has been classified)

$$s(i) = \begin{cases} \frac{b(i) - a(i)}{\max(a(i), b(i))}, & \text{if } |C_i| > 1 \\ 0, & \text{if } |C_i| = 1 \end{cases}$$

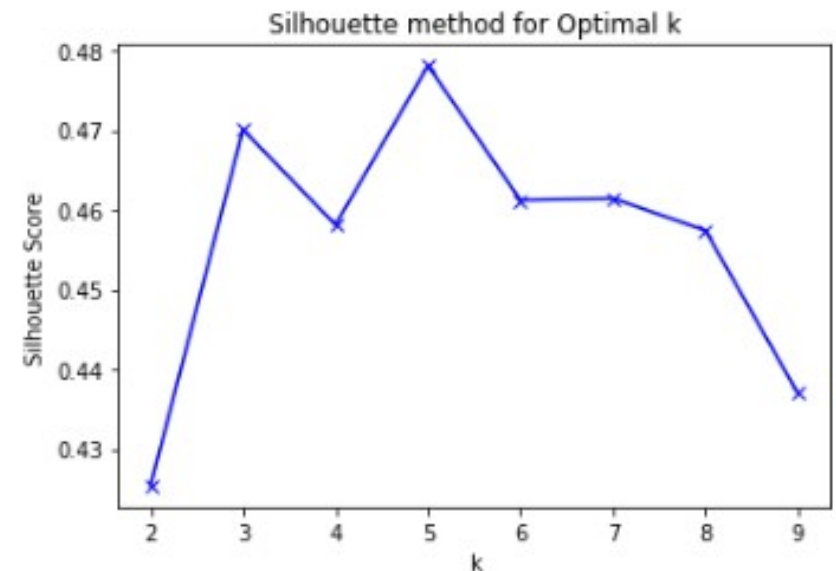
- $a(i)$ is the measure of dissimilarity of element i to its own cluster (where C_i is the set of points in the same cluster as i)

$$\forall i \in C_i, a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

- $b(i)$ dissimilarity to points in other clusters

$$\forall i \in C_i, b(i) = \min_{j \neq i} \frac{1}{|C_j|} \sum_{k \in C_j} d(i, k)$$

- Score for point in center of cluster is close to 1
for point in between clusters is close to 0
for point in wrong cluster is close to -1
- We compute average silhouette score for different cluster counts k



K-Medoids

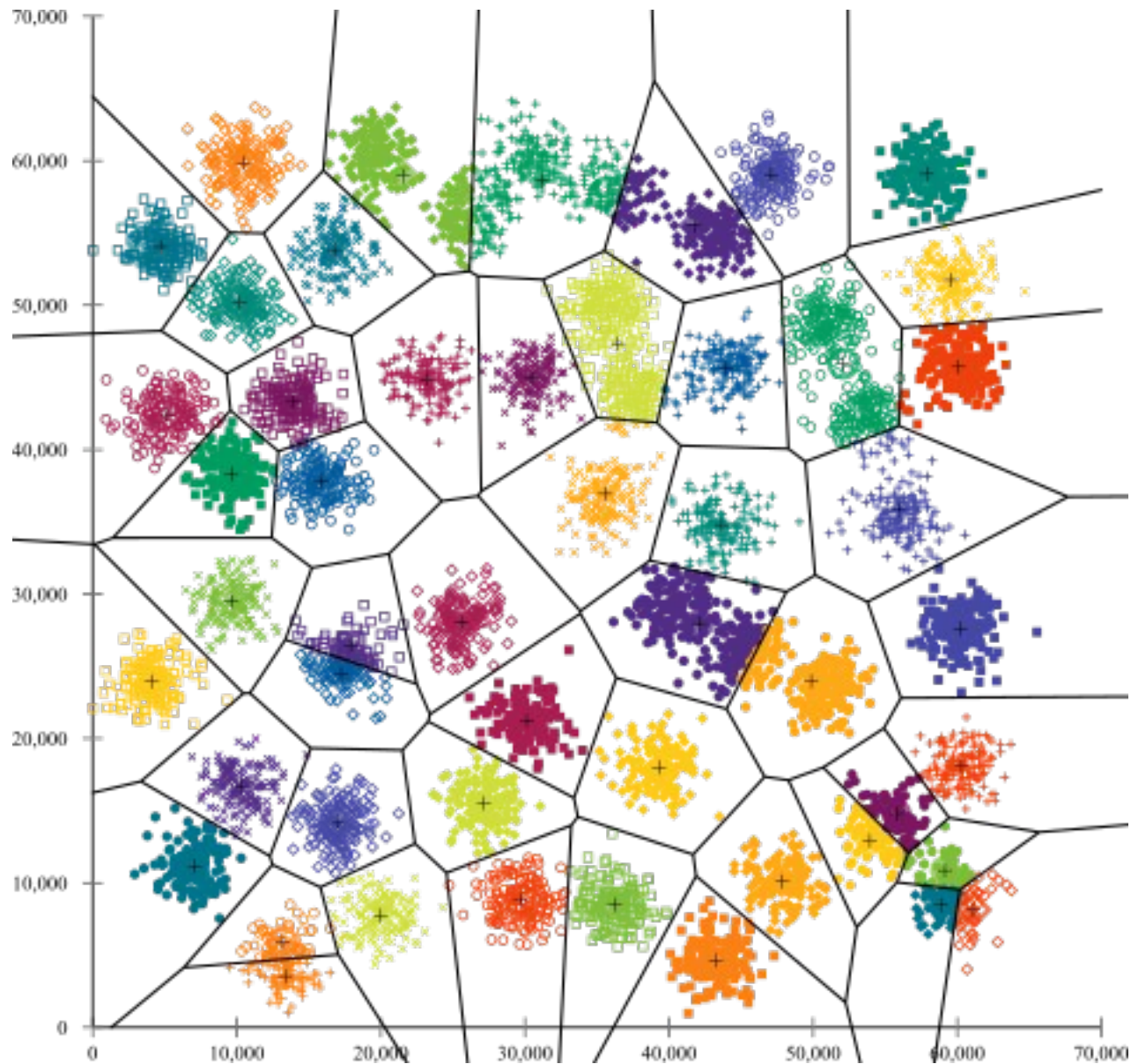
- Does central point always exist? Is it consistent with the distance measure?
 - e.g. only points on a grid allowed
 - e.g. no average text document

medoid of a cluster = the object in the cluster with minimal average dissimilarity to all the objects in the cluster

Observations

- K-means = at some time top 10 data mining algorithm
 - Problems?
 - How to distribute?

Local optimum example



Clustering big data

Big data = too big/slow to process on a single
computer

How can we
distribute K-means?

Observations

- K-means = at some time top 10 data mining algorithm
 - No theoretical guarantees
 - Mediocre practical performance
 - Lots of iterations to converge
 - Sensitive to initialization and outliers
 - Can get stuck on local optimum (if multiple initial centers in the same cluster)
 - Not efficient to distribute (lots of iterations)
- Improvements (++ , ||)
 - Theoretical guarantees
 - Good practical performance
 - Few iterations to converge
 - Improved initialization
 - Efficient to distribute (||)

Spark MLlib

- Contains only algorithms intended to run well on a cluster
 - Contains “novel” algorithms
 - Can be used to process large datasets
 - For single node processing use scikit-learn, Weka or Elki
 - For single core linear algebra use Breeze or MTJ
 - To distribute you can also check: DASK (NumPy, pandas, scikit-learn) or Koalas (pandas on Spark)
- From Spark 2 based on DataSets/DataFrames
 - Implementations are rewritten
 - Watch out which packages you import
- Remember to
 - Prepare input data
 - Scale (StandardScaler)
 - Featurize (use NLTK library and TF-IDF)
 - Label (MLlib requires 0..C-1)
 - Cache
 - Check if default configuration parameters are OK, e.g. number of iterations

K-means in MLlib

K-means++

- David Arthur and Sergei Vassilvitskii. 2007.

K-means++: the advantages of careful seeding.

In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '07). Society for Industrial and Applied Mathematics, USA, 1027–1035.

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 $\frac{d^2(x, \mathcal{C})}{\phi_X(\mathcal{C})}$

4: $\mathcal{C} \leftarrow \mathcal{C} \cup \{x\}$

5: **end while**

$$\Phi_Y(C) = \sum_{y \in Y} d^2(y, C)$$

- (8 log k)-approximation
 - running Lloyd's iteration on top of this will only improve the solution, but no guarantees can be made
- Difficult to distribute
 - Requires k passes over whole data, k can be large (e.g. 1000 types of users of netflix), doesn't scale
 - Idea: pick points independently, oversample, each step pick many points from current distribution, recluster to pick k centers from the sample
- Picking outliers at initialization can confuse it

K-means||

- Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. 2012. **Scalable k-means++**. Proc. VLDB Endow. 5, 7 (March 2012), 622–633. <http://arxiv.org/abs/1203.6402>
- Oversample with factor l ($l = \Theta(k)$); do much less than passes than k-means++
- Choose $l > 1$ and some initial C , then sample points for some iterations independently and add to C , then cluster C and pick centers

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: $\mathcal{C}' \leftarrow$ sample each point $x \in X$ independently with probability $p_x = \frac{\ell \cdot d^2(x, \mathcal{C})}{\phi_X(\mathcal{C})}$
 - 5: $\mathcal{C} \leftarrow \mathcal{C} \cup \mathcal{C}'$
 - 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
-

grow C some number of times

Use centralized k-means++.

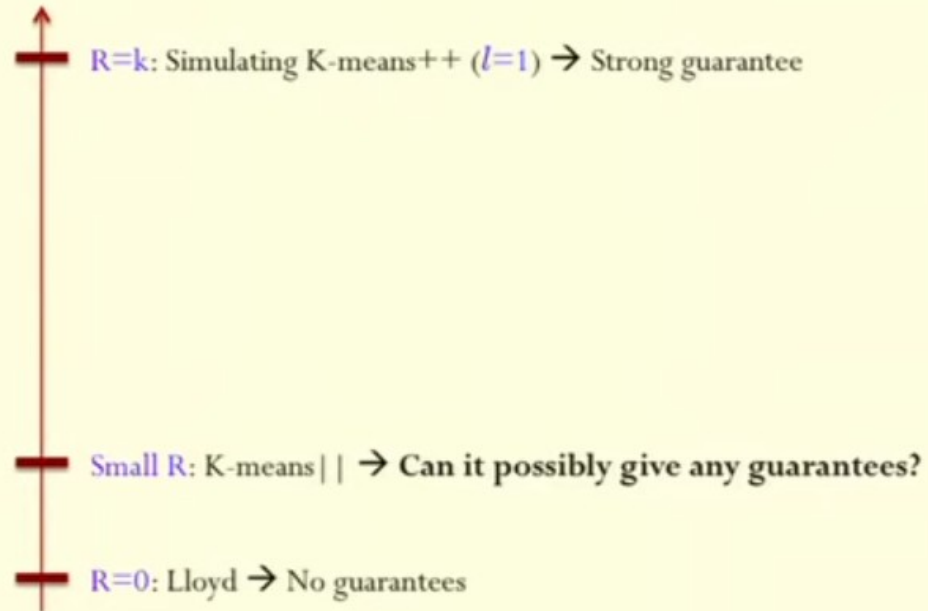
Weights are used to increase probability of selecting as initial center.

- Number of initialization rounds can be configured to much less than $\log(\Psi) \leq \log(n^2 \min_{x, y \in X} d(x, y)^2)$

K-means | |: Intuition

- An interpolation between Lloyd and K-means++

Number of
iterations (R)



Advantages

- Reduces number of Lloyd iterations even more than k-means++

	$k = 20$	$k = 50$	$k = 100$
Random	176.4	166.8	60.4
k -means++	38.3	42.2	36.6
k -means $\ell = 0.5k, r = 5$	36.9	30.8	30.2
k -means $\ell = 2k, r = 5$	23.3	28.1	29.7

Table 6: Number of Lloyd's iterations till convergence (averaged over 10 runs) for SPAM.

- Even without Lloyd iterations usually better than k-means++ with them
 - Because outliers in the initial sample (which would be picked since they have higher probability) don't matter so much

K-means|| results (cont)

	$k = 20$		$k = 50$		$k = 100$	
	seed	final	seed	final	seed	final
Random	—	1,528	—	1,488	—	1,384
k -means++	460	233	110	68	40	24
k -means $\ell = k/2, r = 5$	310	241	82	65	29	23
k -means $\ell = 2k, r = 5$	260	234	69	66	24	24

Table 2: The median cost (over 11 runs) on SPAM scaled down by 10^5 . We show both the cost after the initialization step (*seed*) and the final cost after Lloyd's iterations (*final*).

Enough, because outliers
in the initial sample don't
matter so much