Scalable k-means++

Jeffrey Lehmann, Zachary Nawrocki, and Ziping Song

Pedagogical Course Project
Randomized Algorithms for Machine Learning & Optimization

Agenda

- The selected paper
- What is k-means?
- What is k-means++?
- What is k-means|| and how does it work?
- Runtime, tradeoff of quality with runtime
- Comparing the three algorithms via trials

Paper: Scalable K-Means++

- Authors
    - Bahman Bahmani, Stanford University, Stanford, CA
    - Benjamin Moseley, University of Illinois, Urbana, IL
    - Andrea Vattani, University of California, San Diego, CA
    - Ravi Kumar, Yahoo! Research, Sunnyvale, CA
    - Sergei Vassilvitskii, Yahoo! Research, New York, NY
- Date: Thursday, March 29, 2012
- Venue: Proceedings of the VLDB Endowment (PVLDB), Vol. 5, No. 7, pp. 622-633 (2012)
- Problem Addressed: How to drastically reduce the number of passes needed to obtain, in parallel, a good initialization for the K-means algorithm.

What is k-means? - The Foundations and Motivations

- A solution to a problem in unsupervised learning
- Unsupervised learning: A preprocessor to organize the data for supervised learning
- The problem: clustering
    - How can we group data points?

What do we need to figure out, to cluster data?

- How many clusters? **- k**
- What makes a clustering algorithm good?
  - The points in a cluster should be close to each other.
  - Each cluster should be far apart.
  - Efficiency
- How are we going to cluster?
  - Lloyd’s Algorithm

Lloyd's Algorithm

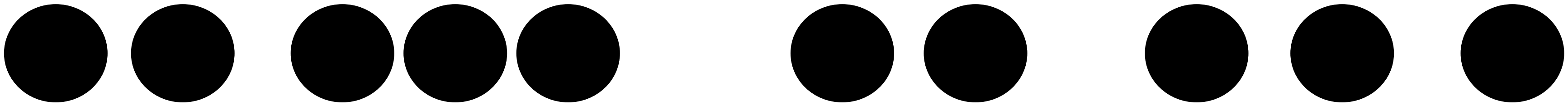
Algorithm 1 Lloyd's Algorithm

$\mu_1, \ldots, \mu_k \leftarrow$ randomly chosen centers
while Objective function still improves **do**
  $S_1, \ldots, S_k \leftarrow \phi$
  for $i \in 1, \ldots, n$ **do**
    $j \leftarrow \arg\min_{j'} \|x_i - \mu_{j'}\|^2\}$
    add i to S_j
  end for
  for $j \in 1, \ldots, k$ **do**
    $\mu_j = \frac{1}{|S_j|} \sum_{i \in S_j} x_i$
  end for
end while

Source: cs.yale.edu/homes/el327/datamining2013aFiles/10_k_means_clustering.pdf

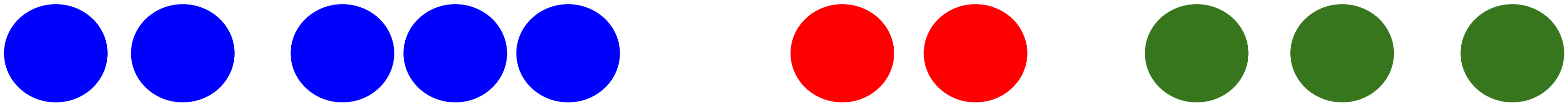
k-means Example, k=3

Visually, we have an idea of a clustering.



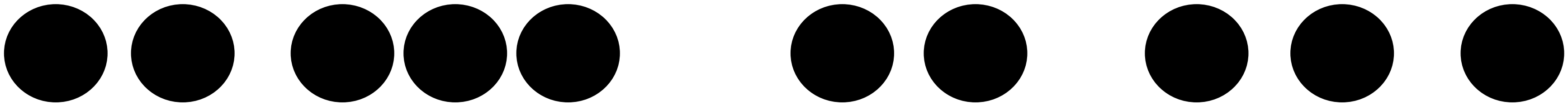
k-means Example, k=3

For example:



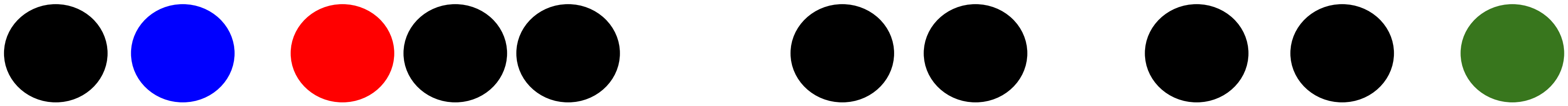
k-means Example, k=3

Now let’s try clustering using Lloyd’s algorithm.



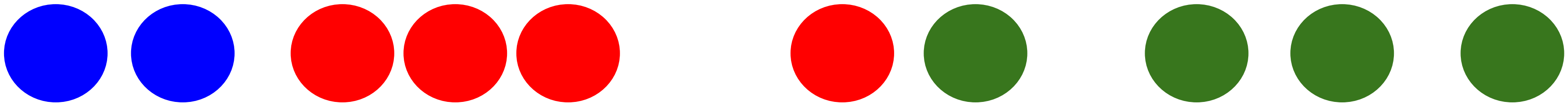
k-means Example, k=3

First, we randomly select k centers. Here, we are using Google’s random number generator.



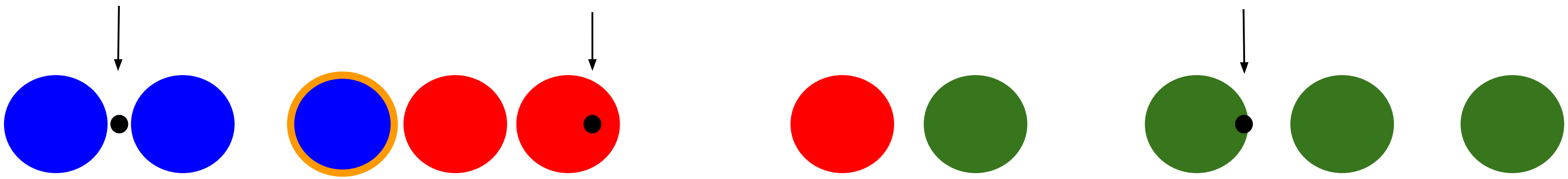
k-means Example, k=3

Next, we group the data points into k clusters, based on their distances to the centers.



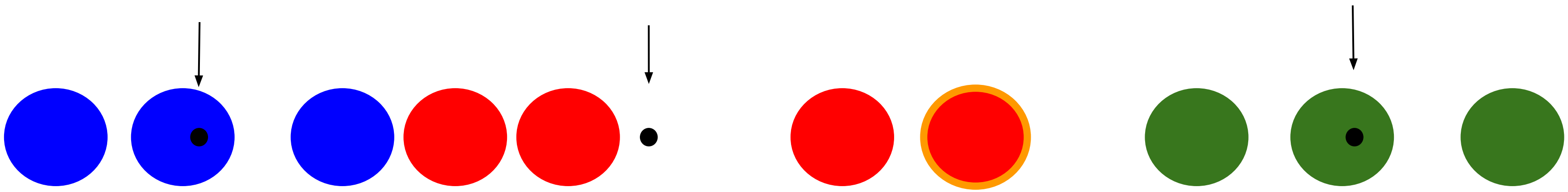
k-means Example, k=3

Then we calculate the mean of each cluster, make it the center, and recluster.



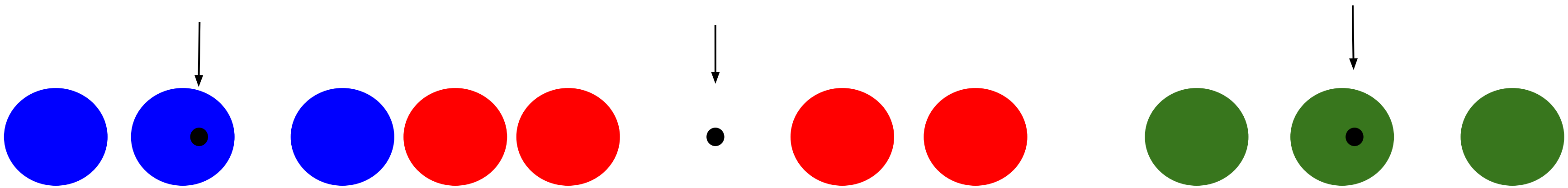
k-means Example, k=3

The clusters were modified, so we repeat the process. Here are the new means and clusters.



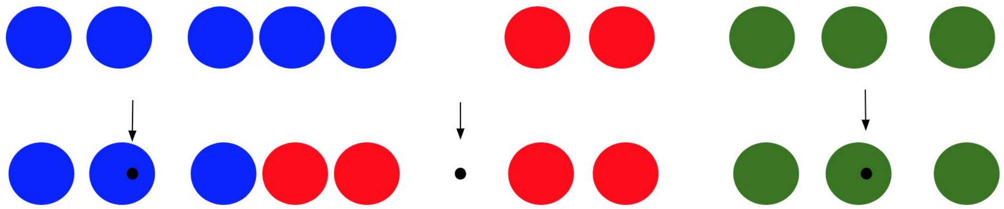
k-means Example, k=3

In this iteration, however, no updates were made. Therefore, we are done with the algorithm.



The Problems with k-means

- Different (and worse) clusterings can result, as seen in the previous example.



- Due to randomization, it is likely that we initialize multiple centers in the same cluster.
- It can take a while to converge, especially with large datasets.
- Lloyd’s Algorithm is NP-hard
- Assuming fixed dimensions, the runtime is O(nkdi), where n is the number of vectors of dimension d, k is the number of clusters, and i is the number of iterations until convergence.
- What would a better algorithm accomplish?

An improvement on k-means: k-means++

- What does it do?
  - Clusters points based off distance to all current clusters
  - Points are chosen with probabilities proportional to their distances from other centers
    - Guarantees clusters are far away
- Is this better than Lloyd’s algorithm?
  - Significantly more accurate
  - Scales better than Lloyd’s (but still not scalable)
    - O(nkd) initialization

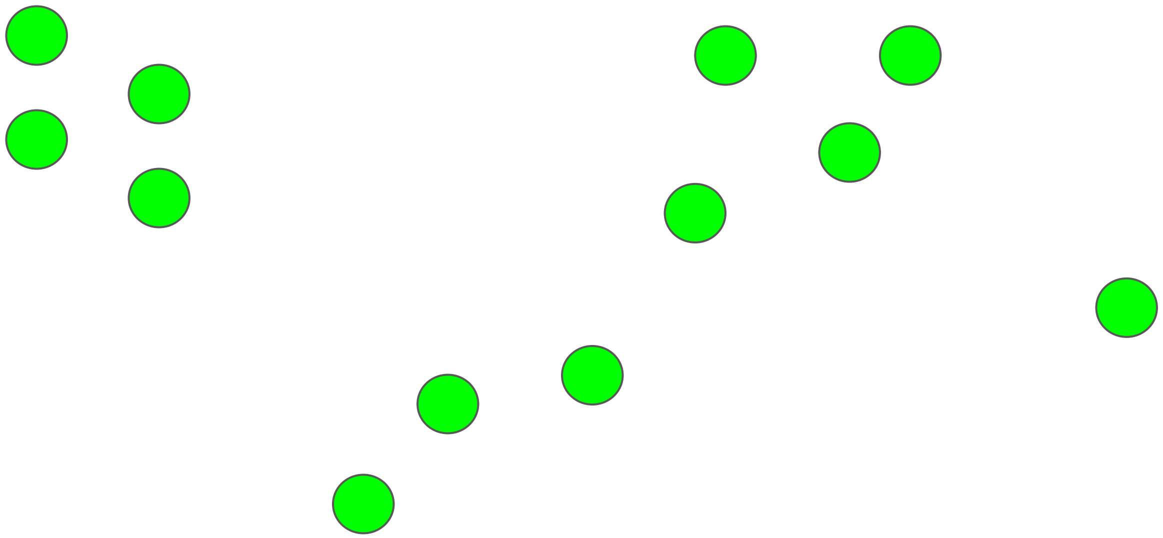
K-means++ Initialization Algorithm

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**

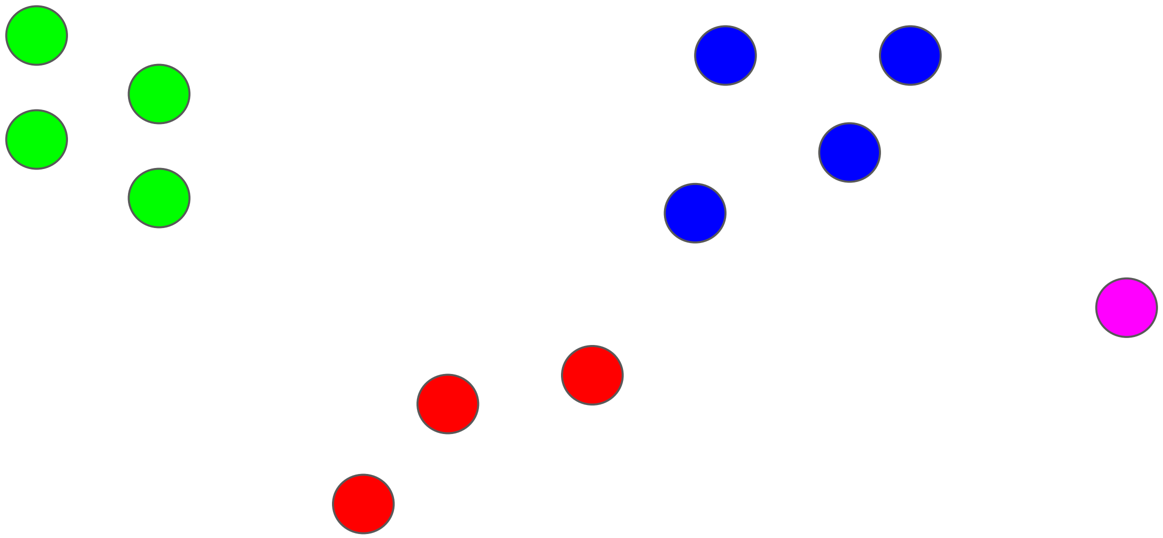
K-means++ example, k = 4

Now let’s try clustering in two dimensions



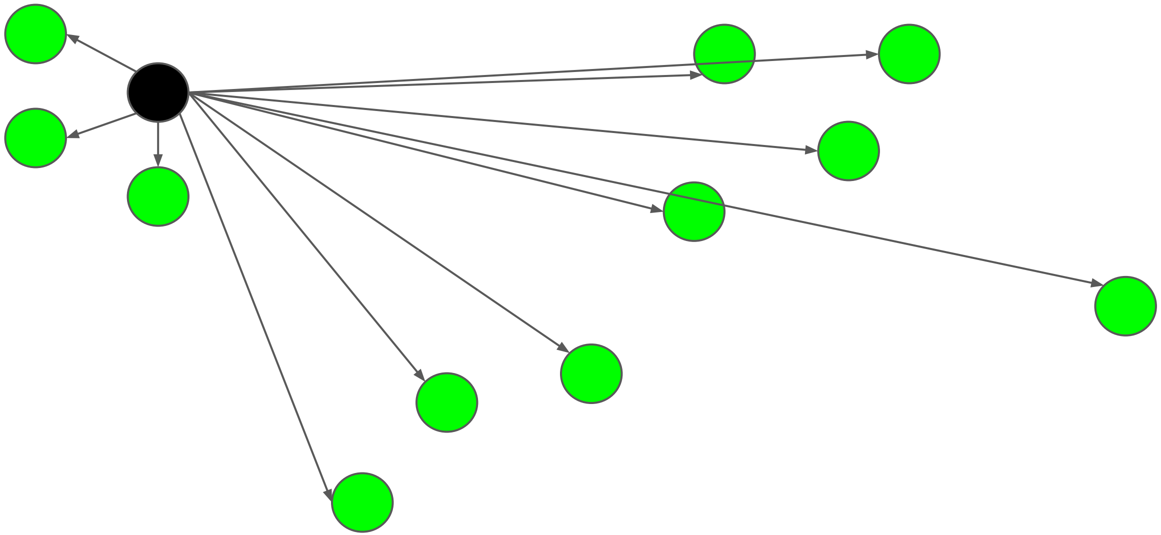
K-means++ example, k = 4

Potential clustering of the data



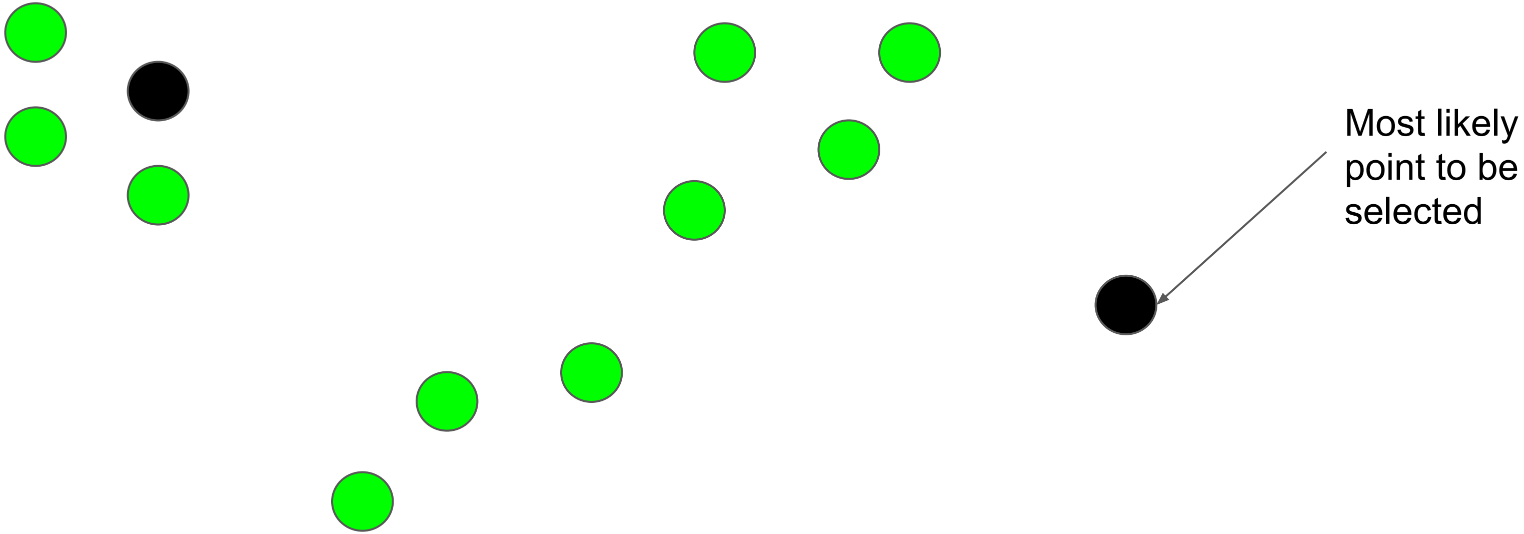
K-means++ example, k = 4

K-means++ starts by picking a random center, then calculates distances



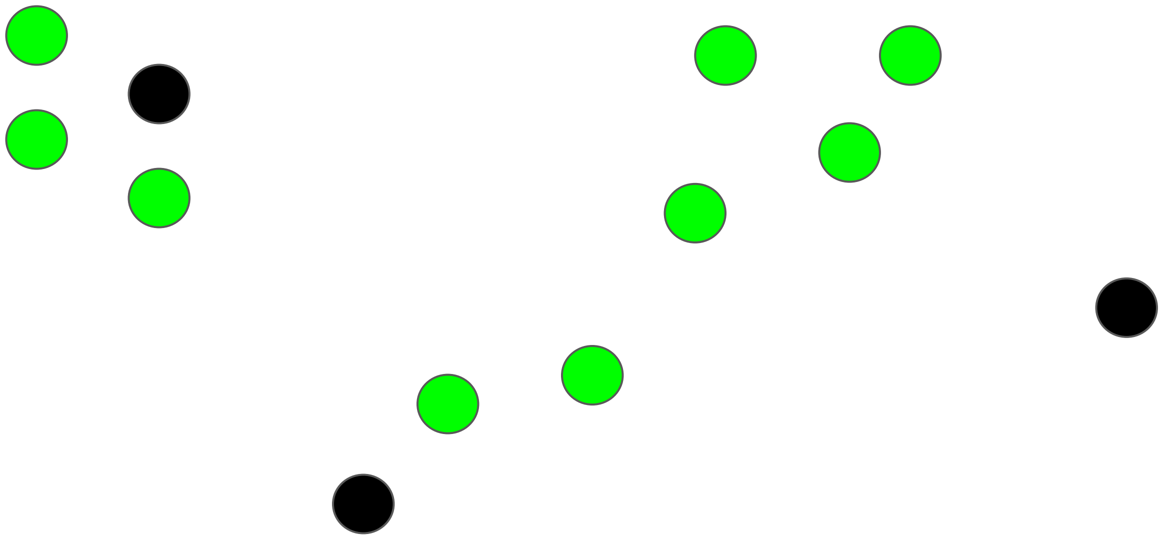
K-means++ example, k = 4

Then it picks another center with probability proportional to distance



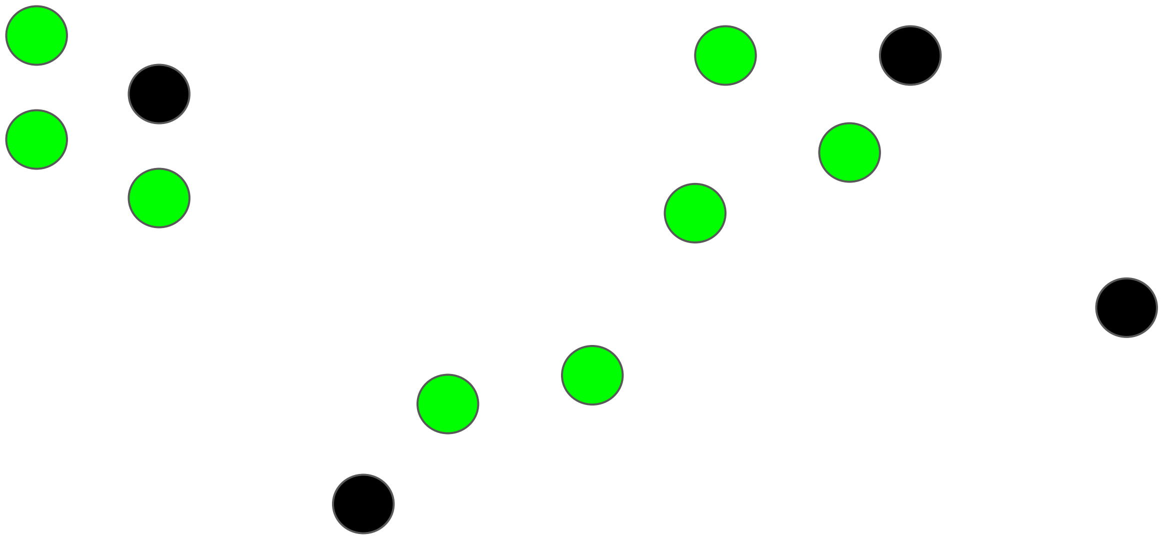
K-means++ example, k = 4

Recalculates distances, and picks new center



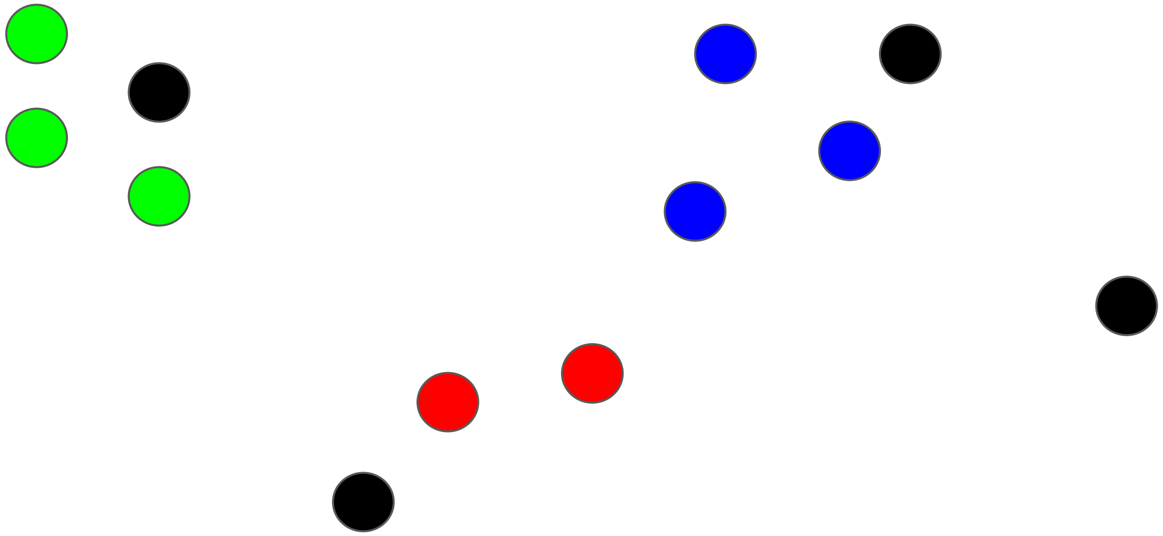
K-means++ example, k = 4

Recalculates the distances to all points from these 3 centers, then picks 4th center



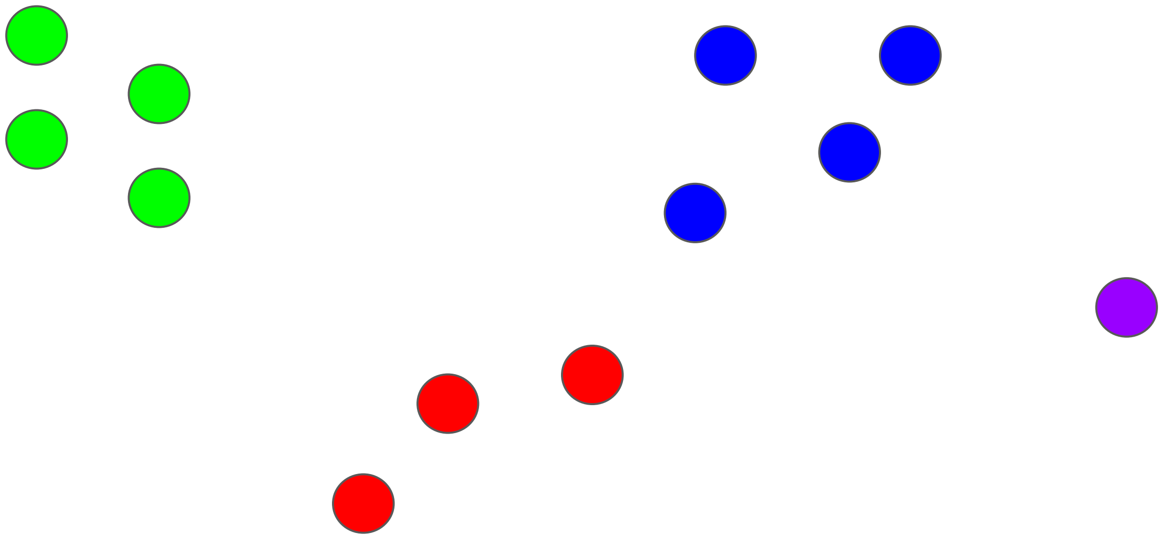
K-means++ example, k = 4

Now that we have k centers we calculate the distance from each point to each center and assign points to cluster with their nearest center.



K-means++ example, k = 4

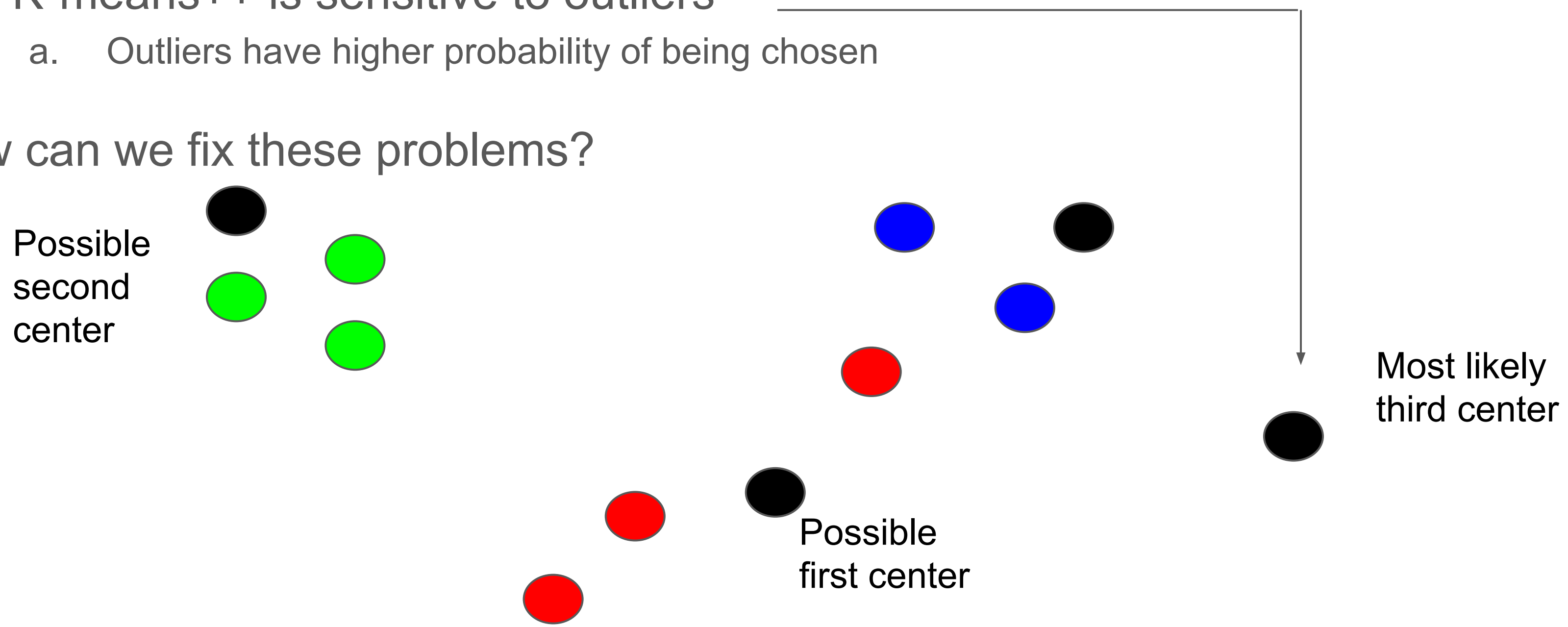
Now k-means++ has finished!



Problems with k-means++

1. K-means++ needs to pass over the data k times
    a. Picks only one center at each iteration
2. K-means++ is sensitive to outliers
    a. Outliers have higher probability of being chosen

How can we fix these problems?



The fix: Scalable k-means, or k-means||

Essential changes:

1. Oversampling factor l
   a. l determines the number of points we pick in each round
2. $(log\ \psi)$ iterations
   a. Depends on the clustering time of the first initialization
   b. $(log\ \psi) << k$
   c. $O(ndlog(\psi))$ initialization

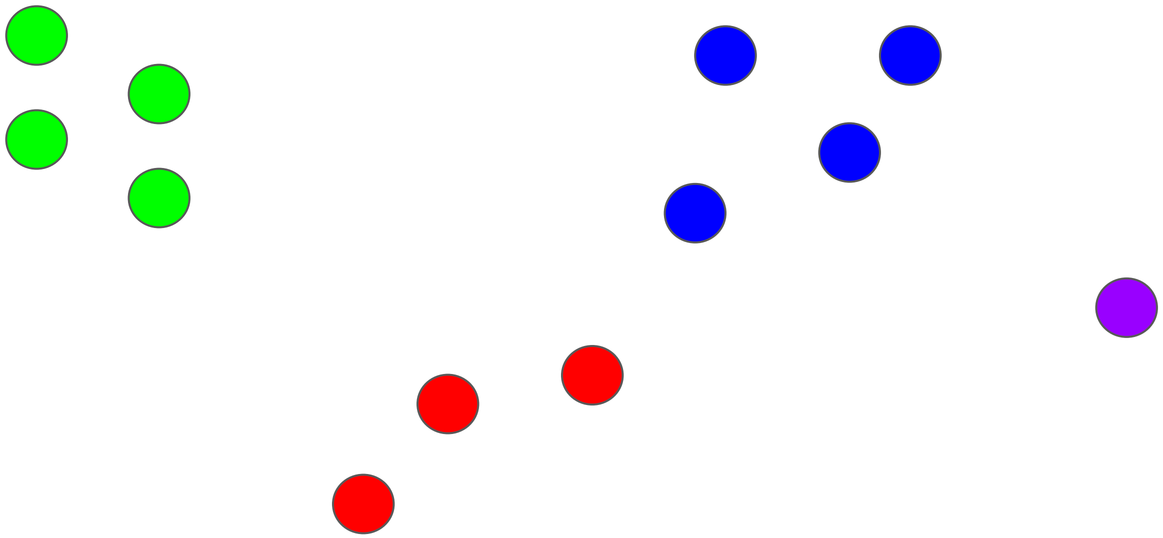
K-means|| algorithm

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

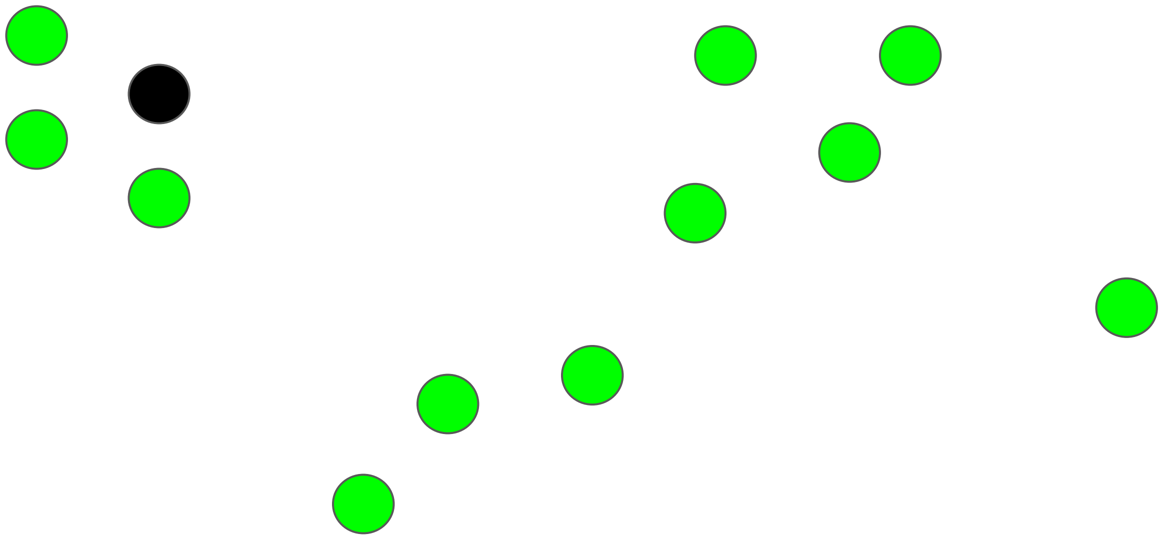
K-means|| example, k = 4, l = 3

Let’s use a familiar example to show the difference between k-means|| and k-means++.



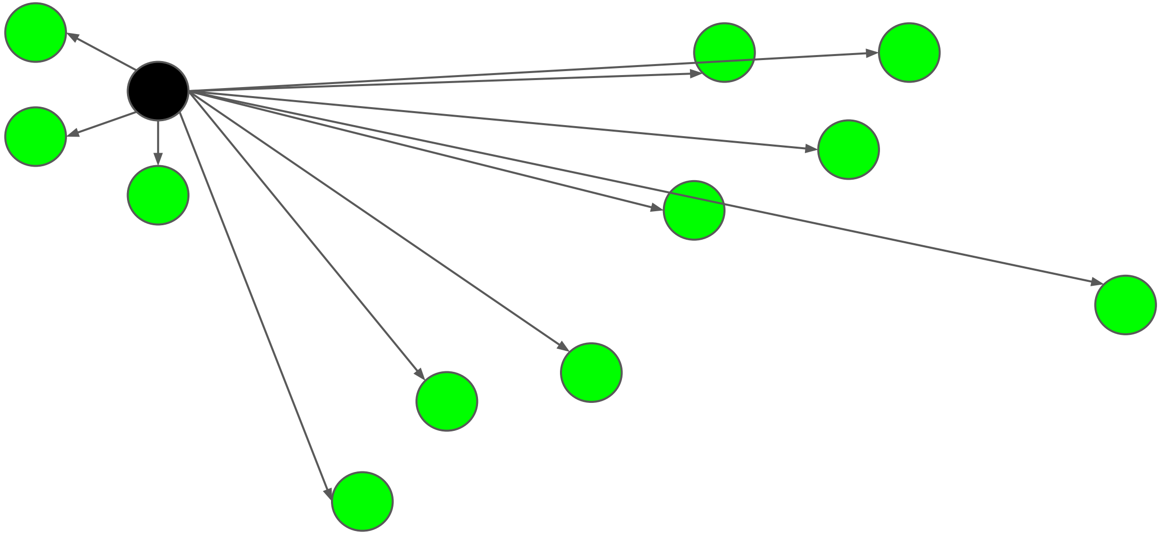
K-means|| example, k = 4, l = 3

Similar to k-means++, k-means|| randomly selects the first center uniformly at random.



K-means|| example, k = 4 l = 3

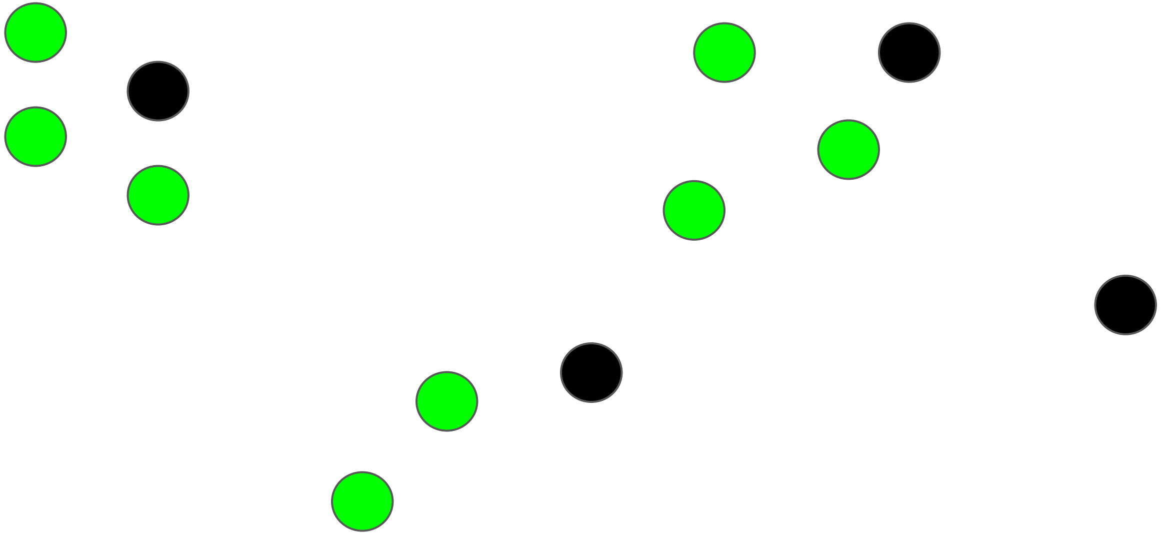
Next we compute the distances to all other points and save the cost of this computation in ψ, which determines the number of iterations we perform.



Note: $\psi \leq n^2d^2$

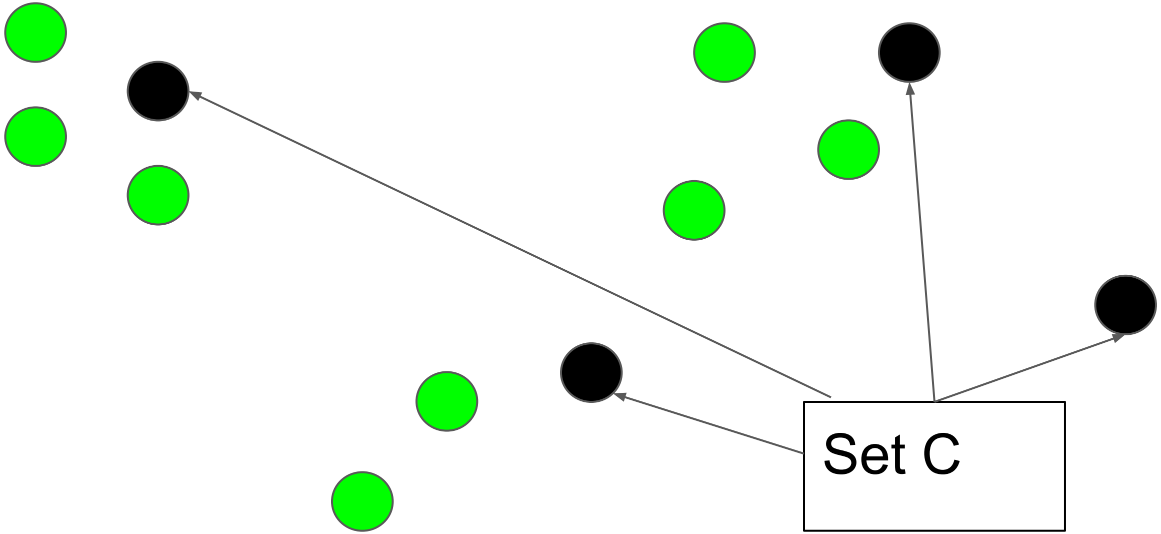
K-means|| example, k = 4, l = 3

Now we do not pick 1 new center (as k-means++ would), but we rather pick l points, in this case 3, with probability proportional to distance.



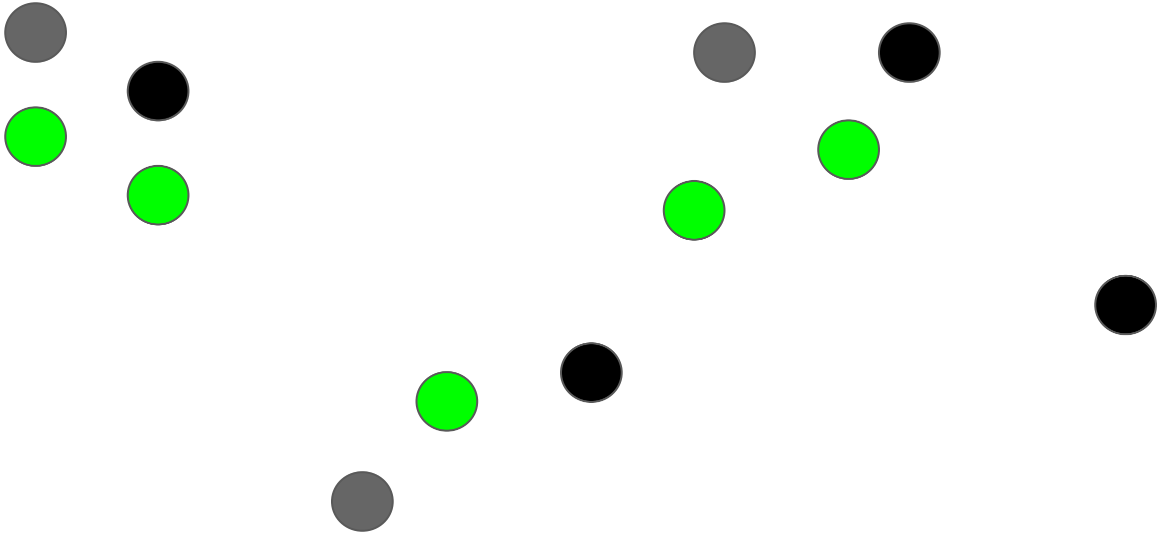
K-means|| example, k = 4, l = 3

Then we compute the distances between all points and these centers. K-means++ would stop here but k-means|| will often calculate more than k centers.



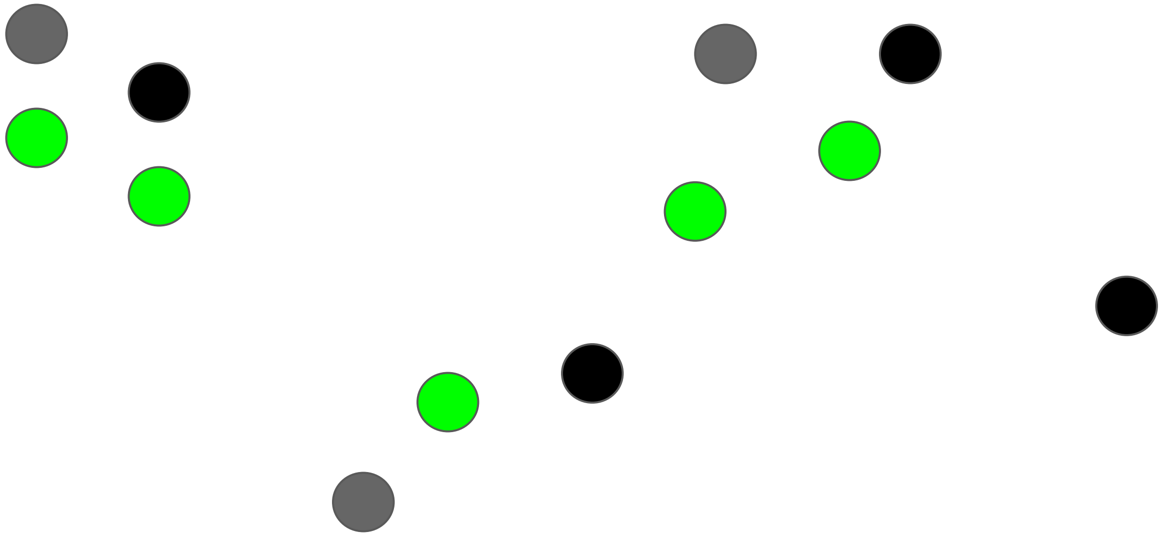
K-means|| example, k = 4, l = 3

So now we pick l more centers and we repeat the process until we have iterated $\log(\psi)$ times.



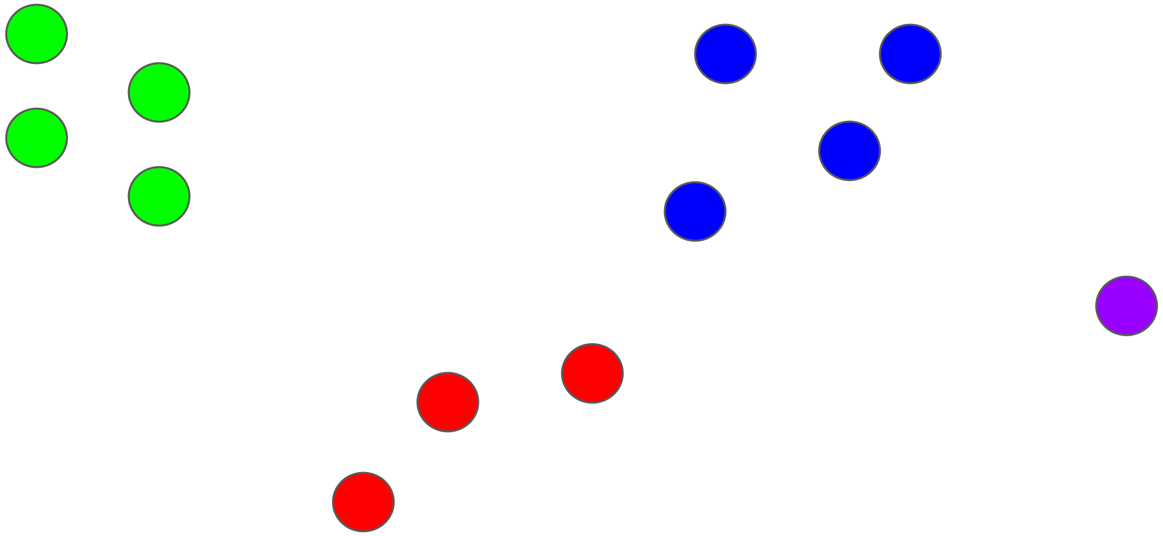
K-means|| example, k = 4, l = 3

Let's say log(ψ) = 2 and that we are done picking centers, we know evaluate the set C containing the 7 selected centers and recluster those to get k centers.



K-means|| example, k = 4, l = 3

We can use Lloyd's algorithm or k-means++ to recluster the 7 centers and get these clusters.



Differences

K-means++	K-means||
K iterations over the dataset	$\log(\psi)$ iterations where $\psi \leq n^2d^2$, scales well
Adds one center per iteration	Adds multiple centers (l centers) per iteration
Calculates exactly k centers	Calculates more than k centers due to oversampling of the data

These runtime differences are nice but with so few iterations being run, can we expect accurate results?

To Do

According to https://www.cs.rpi.edu/~gittea/teaching/fall2020/project.html

- What is the main result of the paper?
- Describe the result or algorithm and motivate it intuitively.
- What is the cost (time, space, or some other metric) of this algorithm, and how does it compare to prior algorithms for the same problem? (and similarly, for non-algorithmic results)
- What performance guarantees, if any, are provided for the algorithm?
- Give an accurate description of the analysis given in the paper: in simple cases this may be a tour through the entire argument; when this is not possible, focus on explaining a core lemma/theorem that supports the claim of the paper.
- Provide an empirical evaluation of the algorithm: compare its performance to reasonable baselines, and explore relevant aspects of the algorithm (its variability, sensitivity to relevant properties of the input, etc.). If presenting a non-algorithmic result and it is possible, provide some experimental evidence of its sharpness or lack thereof.

Experiments

Conclusion/Big Picture

1. K-means|| and k-means++ are accurate
    a. Both find centroids that are better than random initialization
    b. The selected centers are much less likely to be part of the same cluster
2. K-means|| scales well with its few iterations
    a. Scales down the number of iterations on large data sets (i.e., $n > 10^9$) to be magnitudes smaller than k-means++
    b. By decreasing the size of the data, through distribution, problems can be solved more quickly
3. K-means|| is applicable in many real-world unsupervised learning situations.
    a. Whether it's credit fraud detection, organizing featureless data (or data with very few features), or any sort of data analysis, k-means|| can be a useful way for preprocessing the data for supervised learning.

Sources

- **Bahmani, Moseley, Vattani, Kumar, Vassilvitskii, Scalable K-Means++, arXiv:1203.6402 [cs.DB] -- The paper that inspired this presentation**
- Professor Alex Gittens, CSCI 4961, Machine Learning & Optimization Lectures, Rensselaer Polytechnic Institute
- Professor Malik Magdon-Ismail, CSCI 4100, Machine Learning from Data Lectures, Rensselaer Polytechnic Institute
- Seif, The 5 Clustering Algorithms Data Scientists Need to Know, Towards Data Science
- Professor Edo Liberty, Lecture 10: k-means clustering, Algorithms in Data Mining, Yale University
- Arthur, Vassilvitskii, K-means++: The Advantages of Careful Seeding, SODA ‘07