

CS150A Database

Wenjie Wang

School of Information Science and Technology

ShanghaiTech University

Dec. 27, 2024

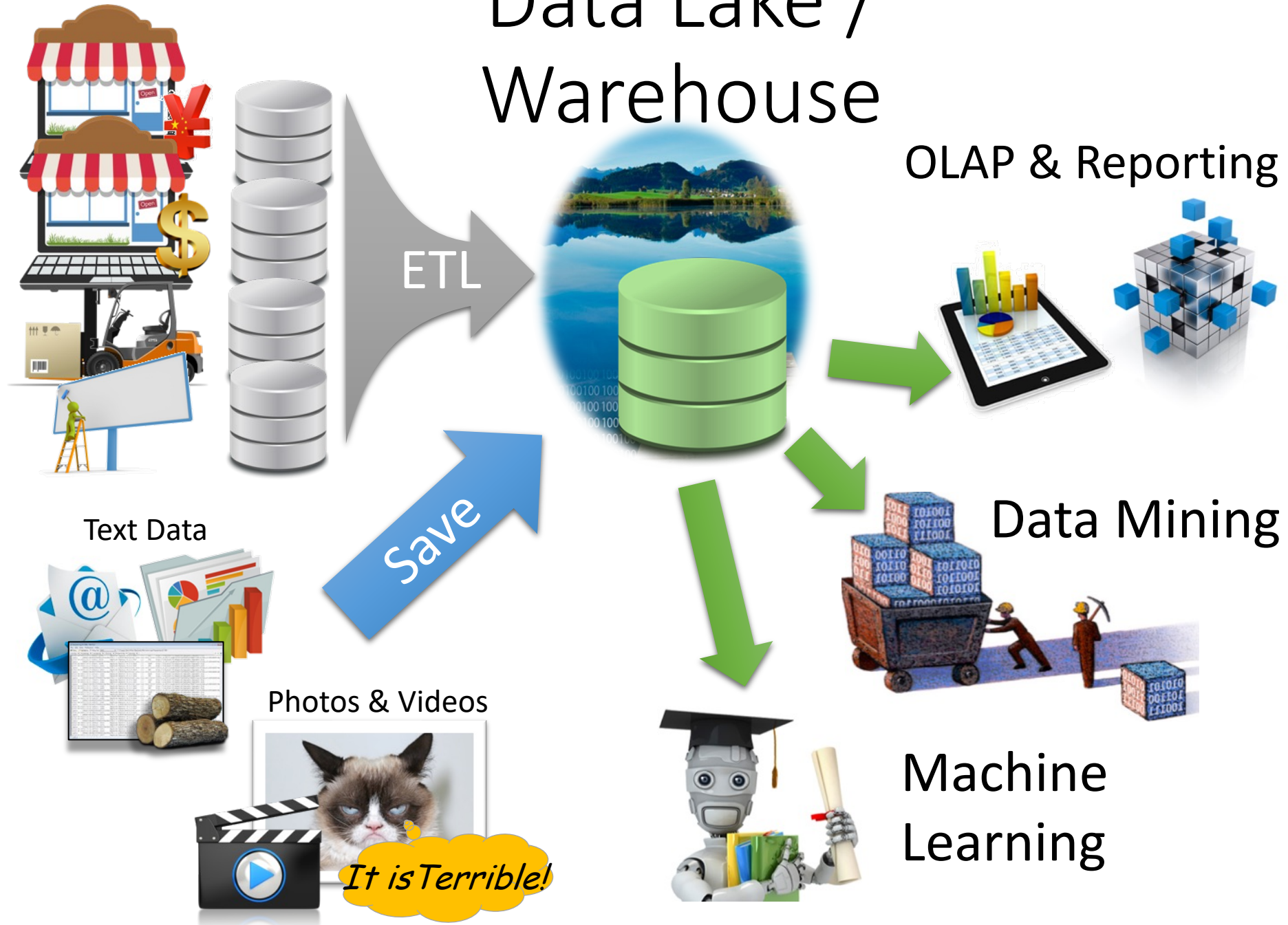
Today:

- Analytics and ML in Data Systems:
 - Part 2
 - Clustering

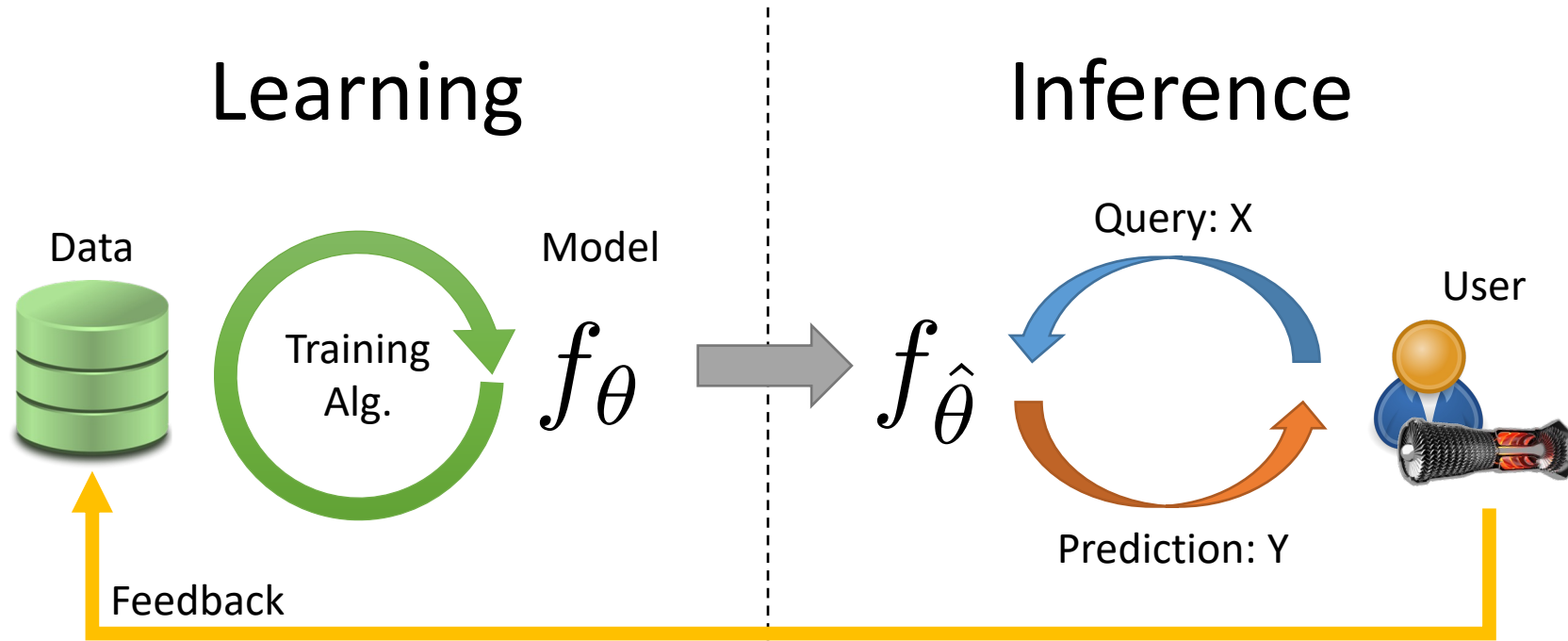
Readings:

- Database Management Systems (DBMS), Chapter 26

Data Lake / Warehouse



Machine Learning Lifecycle

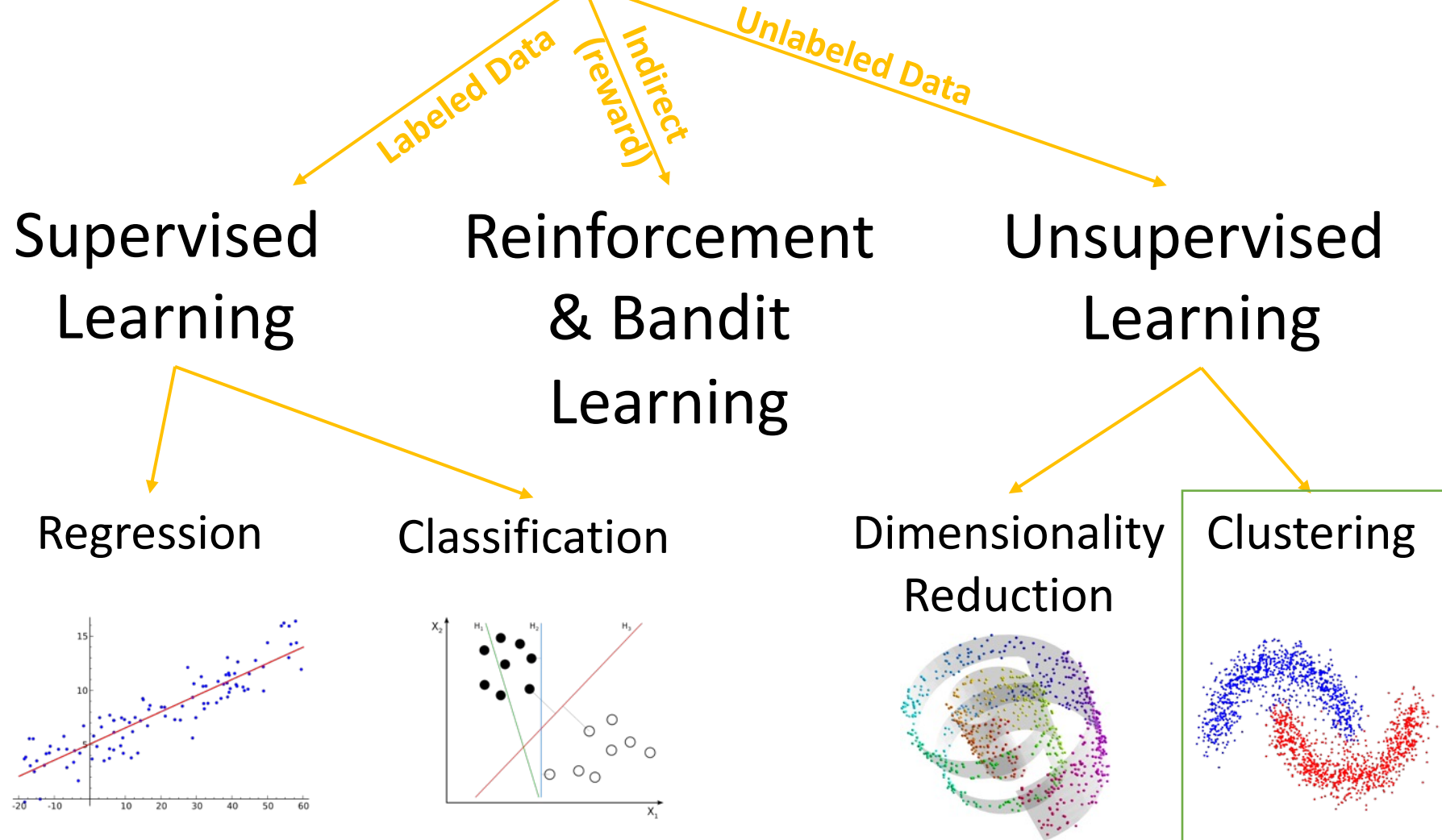


- Typically a time consuming iterative batch process
 - Feature engineering
 - Validation

- Focus is on making fast robust predictions
 - Monitoring and tracking feedback
 - Materialization + fast model inference



Taxonomy of Machine Learning



Clustering Images

- Given a collection of images cluster them into *meaningful groups*.



Clustering Images

- Given a collection of images cluster them into meaningful groups.

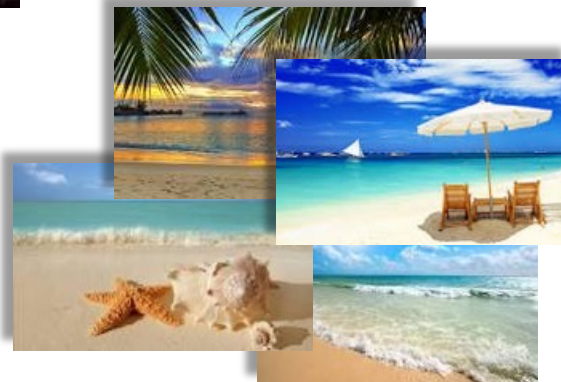
“Mountains”



“Forest”



“Beaches”



Clustering Images

- Given a collection of images cluster them into meaningful groups.



- **Unsupervised:** The labels of the groups are not given in the training data
- **Exploratory:** overlaps with data mining

Clustering Images

- Given a collection of images cluster them into meaningful groups.

Simplified Illustration

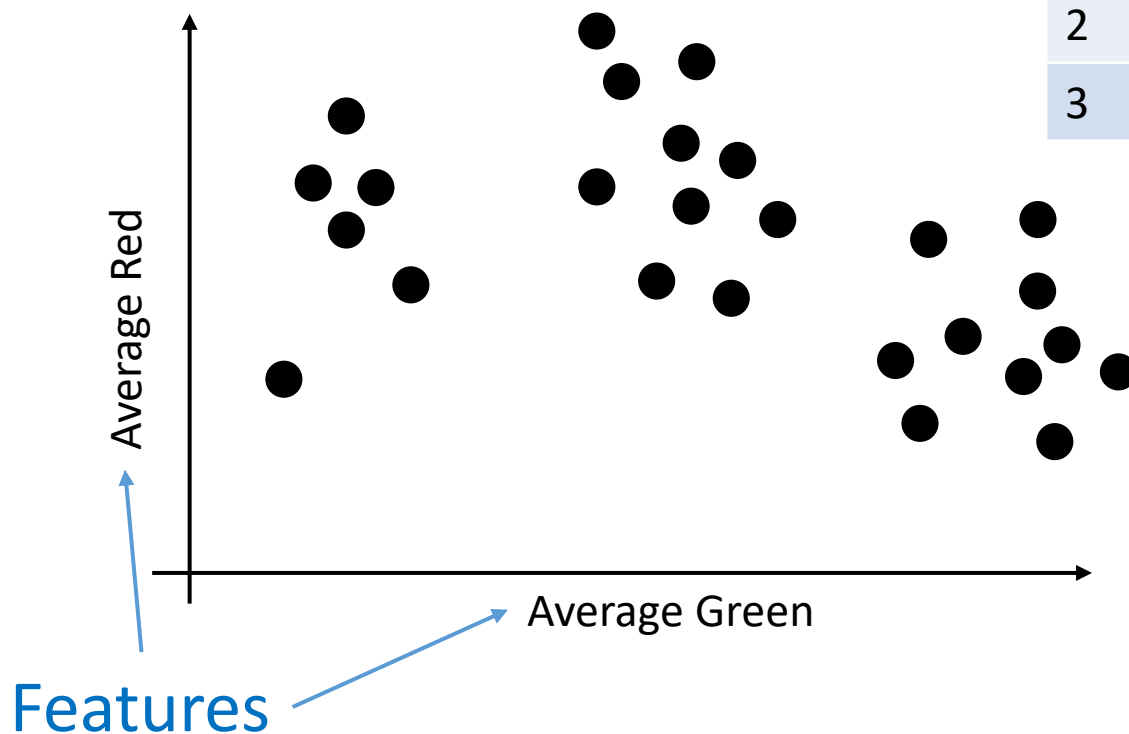


Image Id	Average Red	Average Green
1	123	200
2	212	103
3	55	35

- How many clusters?
- Where are the clusters?

Clustering Images

- Given a collection of images cluster them into meaningful groups.

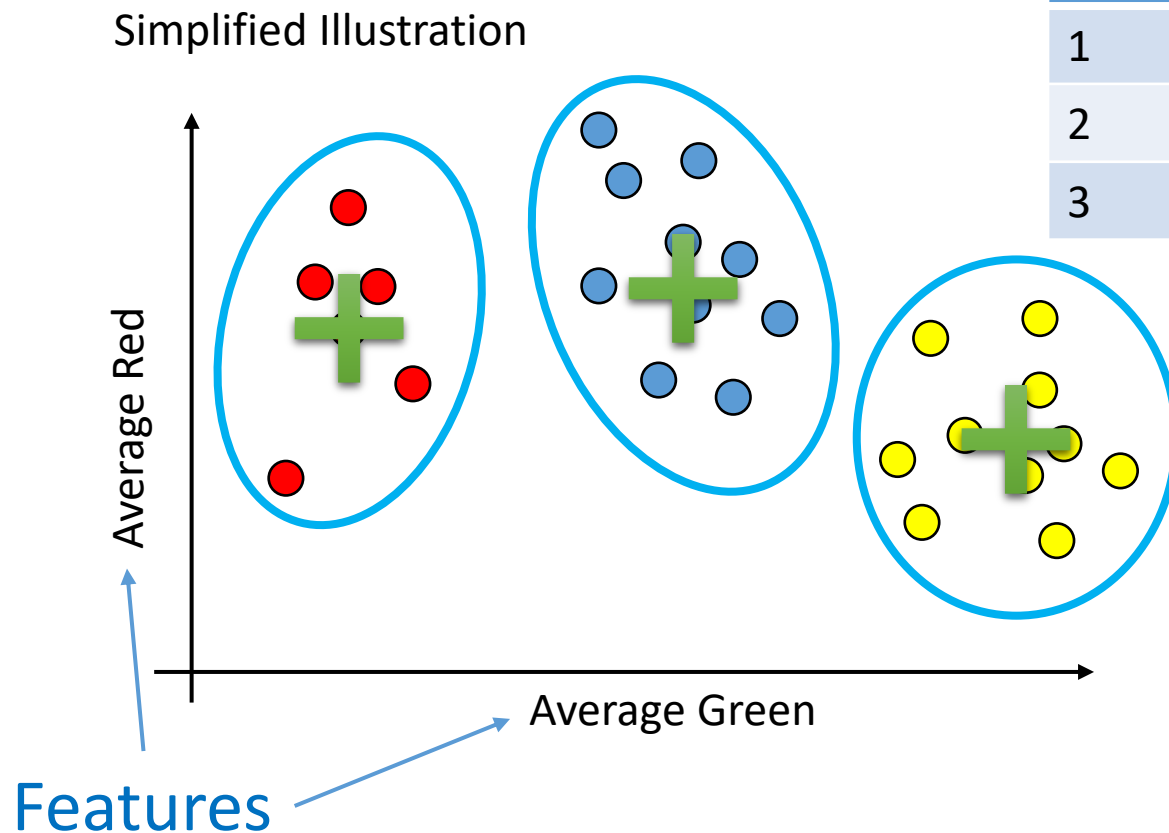
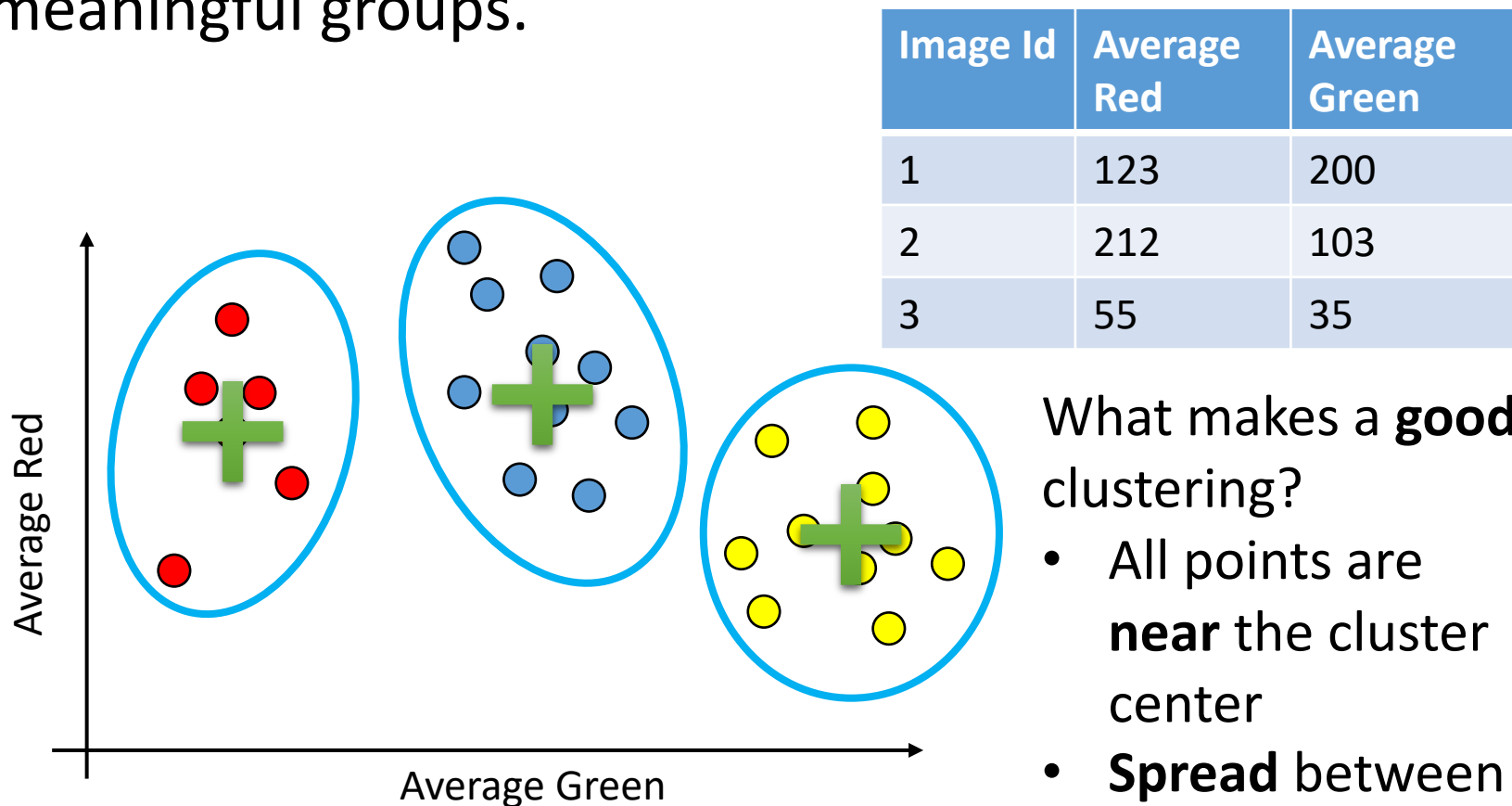


Image Id	Average Red	Average Green
1	123	200
2	212	103
3	55	35

- Where are the clusters?
- How many clusters?

Clustering Images

- Given a collection of images cluster them into meaningful groups.

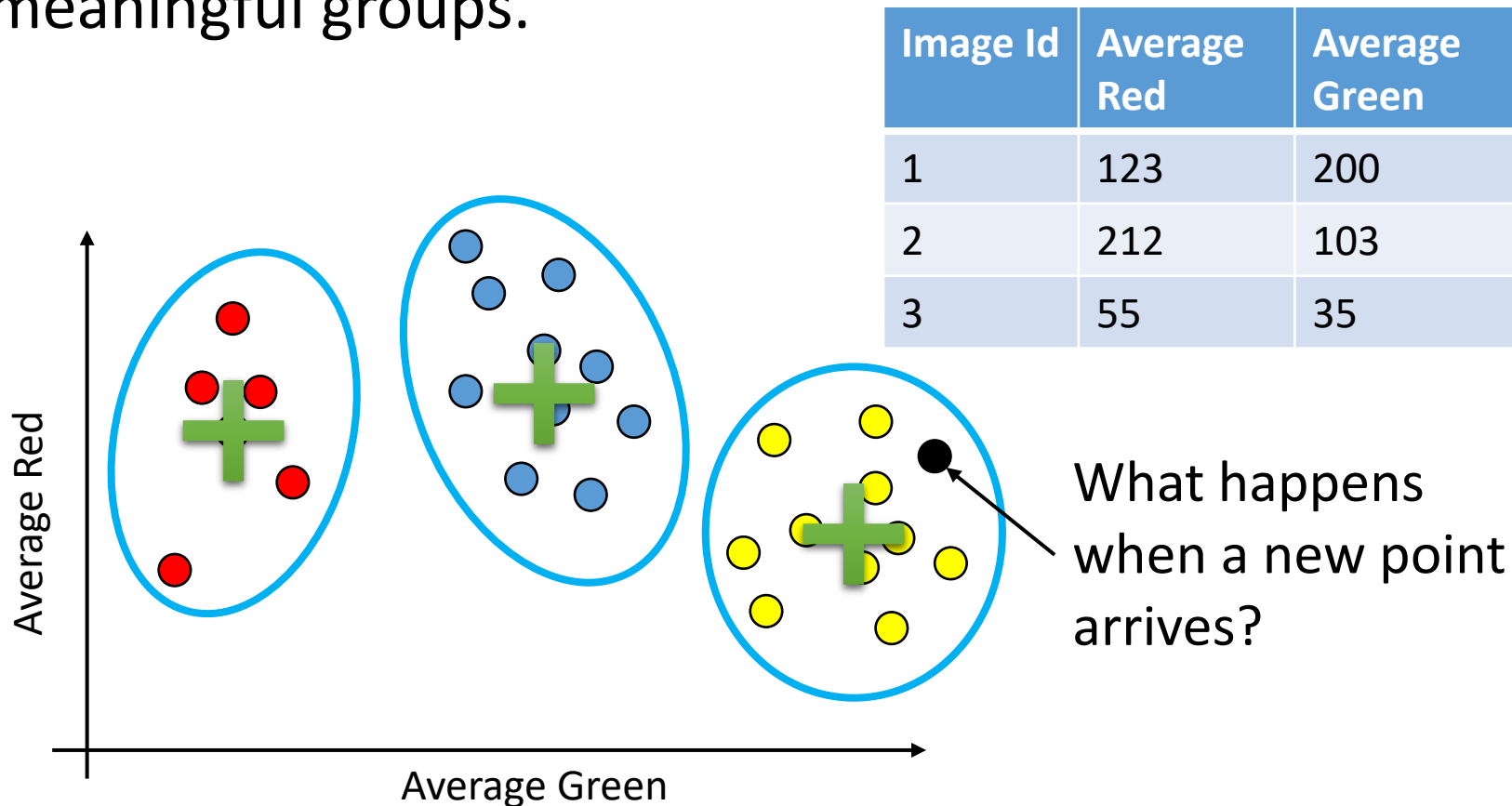


What makes a **good** clustering?

- All points are **near** the cluster center
- Spread** between clusters > spread within clusters

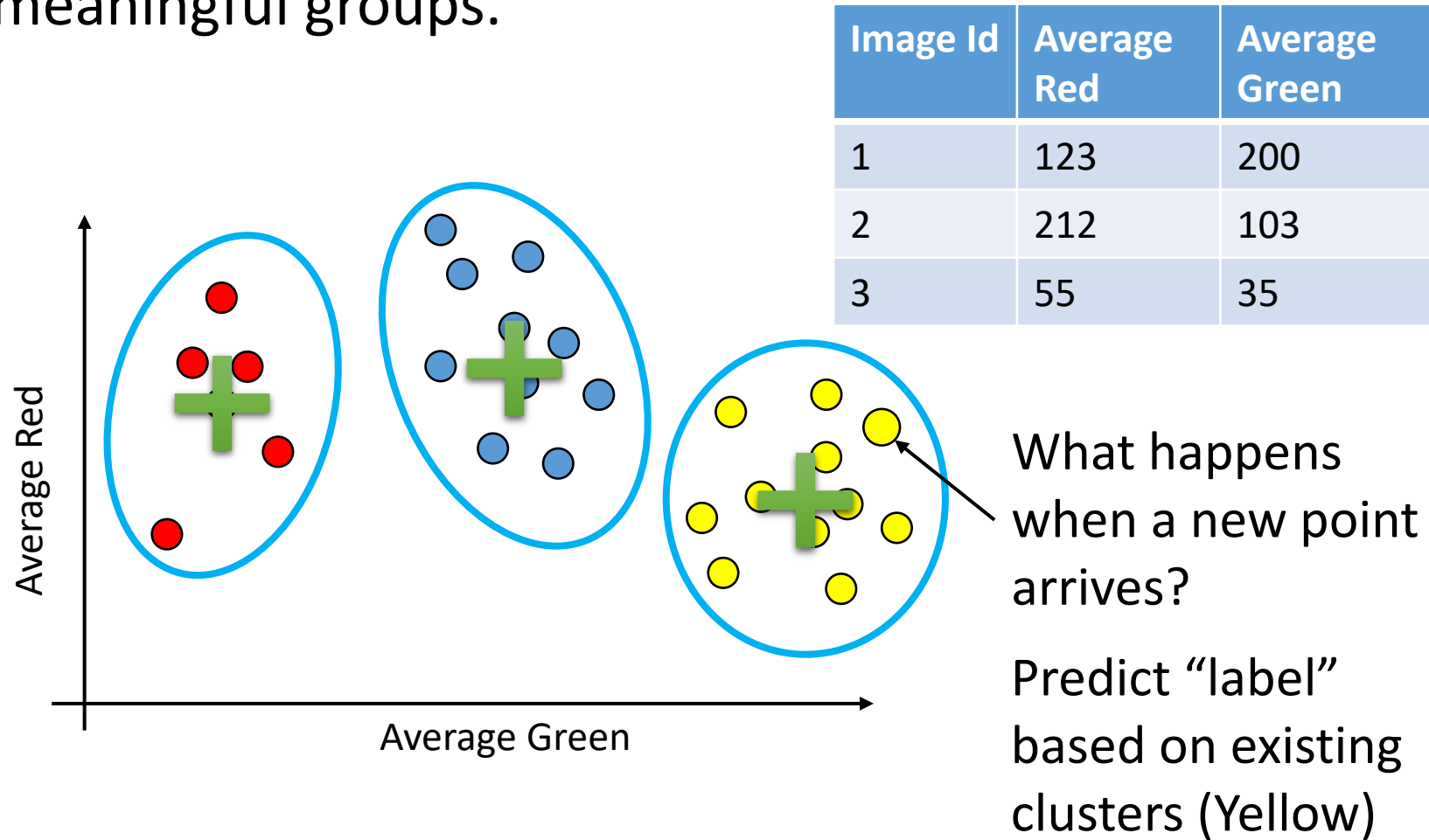
Clustering Images

- Given a collection of images cluster them into meaningful groups.



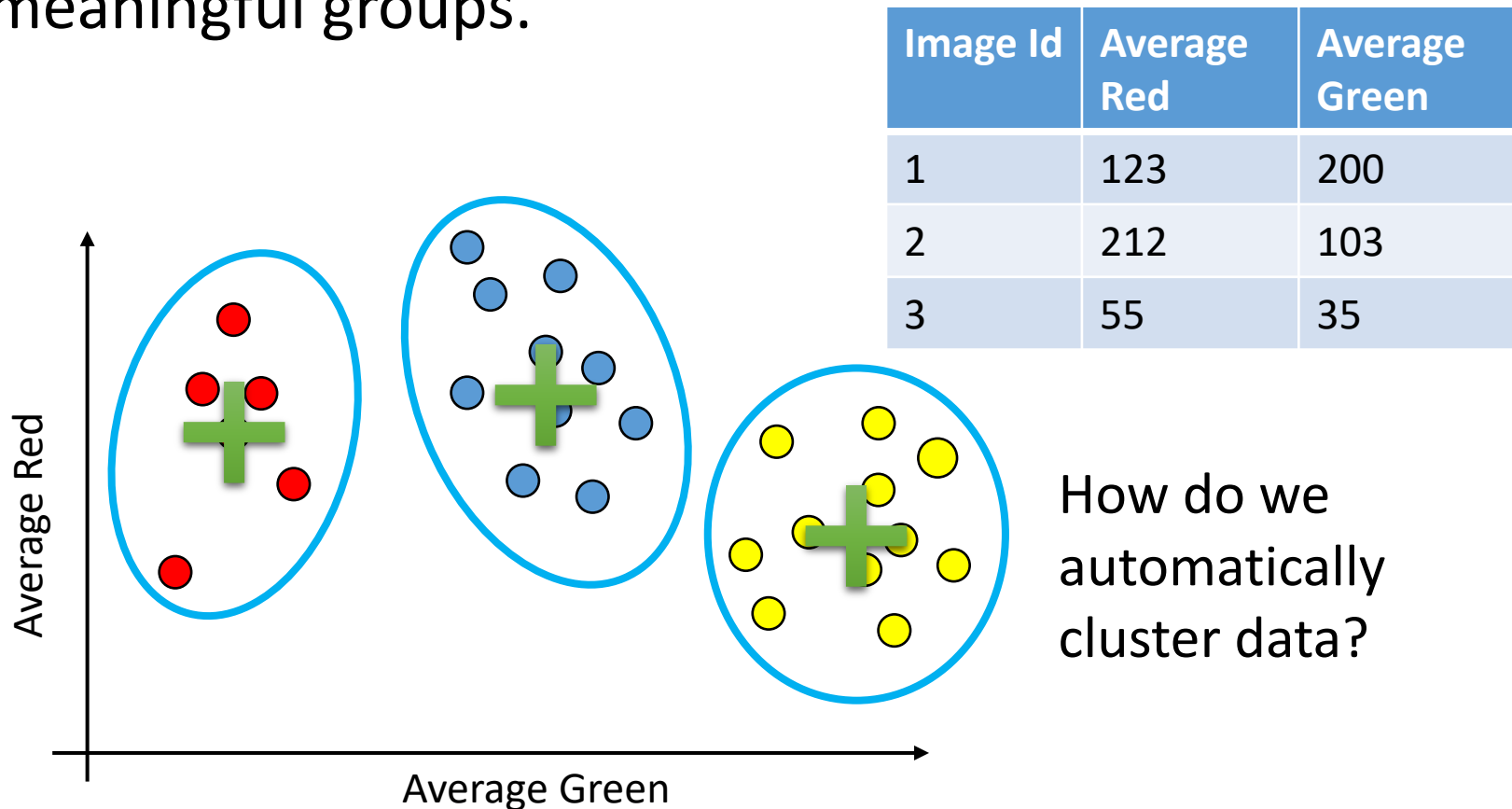
Clustering Images

- Given a collection of images cluster them into meaningful groups.



Clustering Images

- Given a collection of images cluster them into meaningful groups.



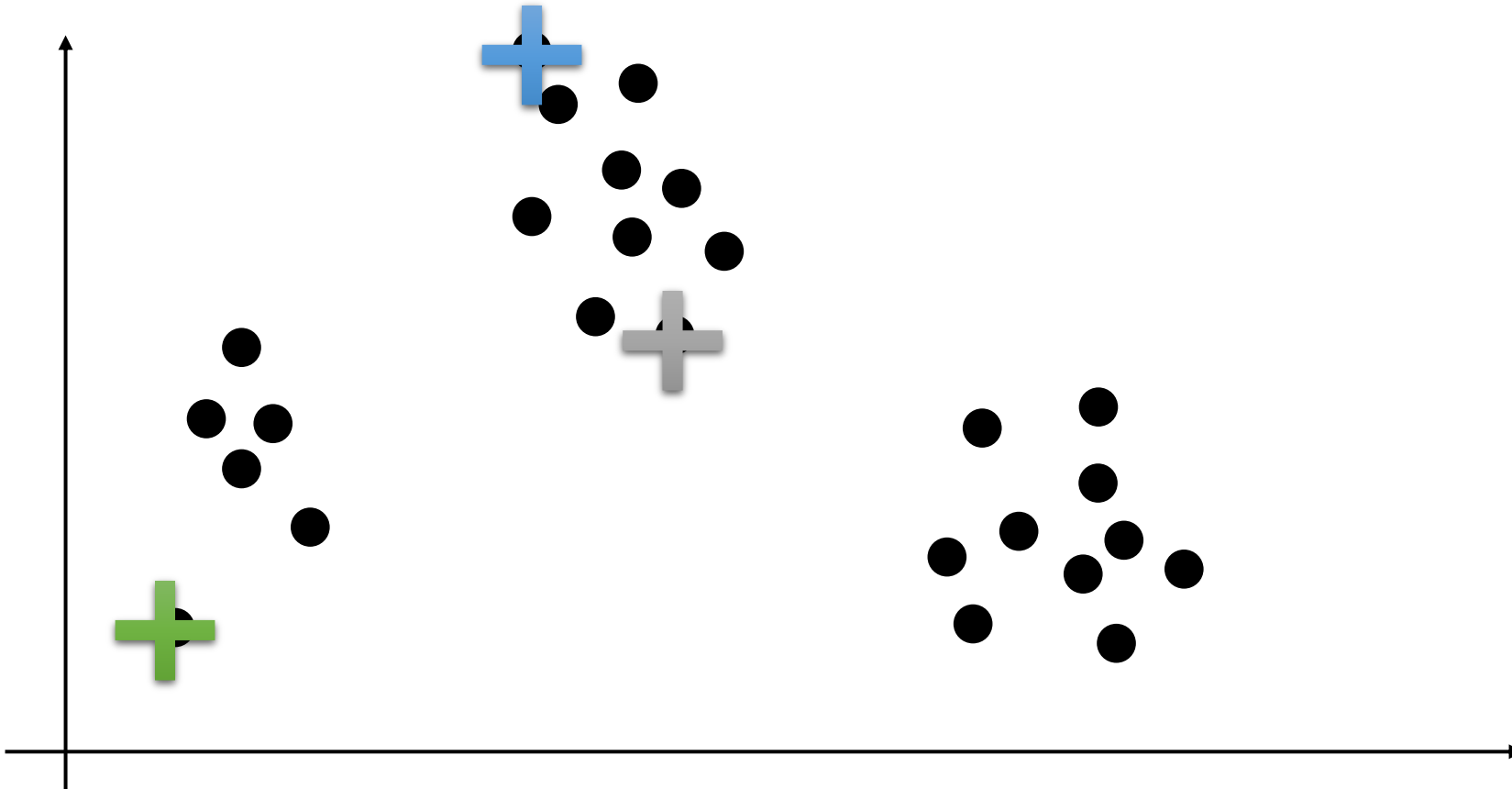
How do we Compute a Clustering?

Many different clustering models and algorithms:

- Feature Based Clustering: *Points in R^d*
 - **K-Means:** EM on Symmetric Gaussians ← We will learn this one
 - **Mixture Models:** Generalized k-means
 - ...
- Spectral Methods: *Similarity Function Between Items*
 - **Similarity based clustering:** *A and B are co-purchased*
 - **Graph clustering:** *Cities based on road network*
 - ...
- Hierarchical Clustering: *clustering nested items*
 - **Latent Dirichlet Allocation:** *Documents based on words*
 - *Developed at Berkeley and widely used!*
 - ...

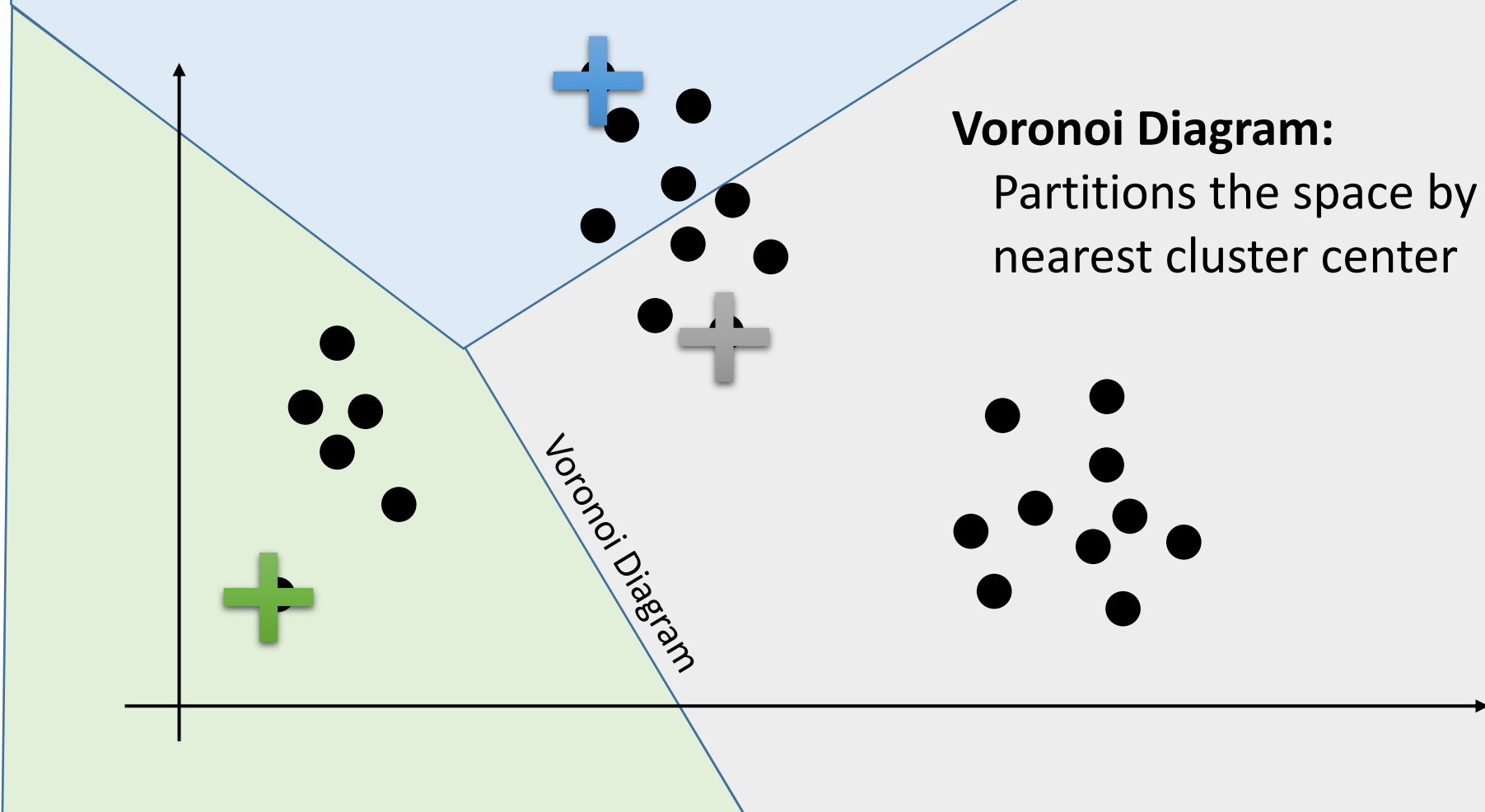
K-Means Clustering: *Intuition*

- Input K: The number of clusters to find
- Pick an initial set of points as cluster centers



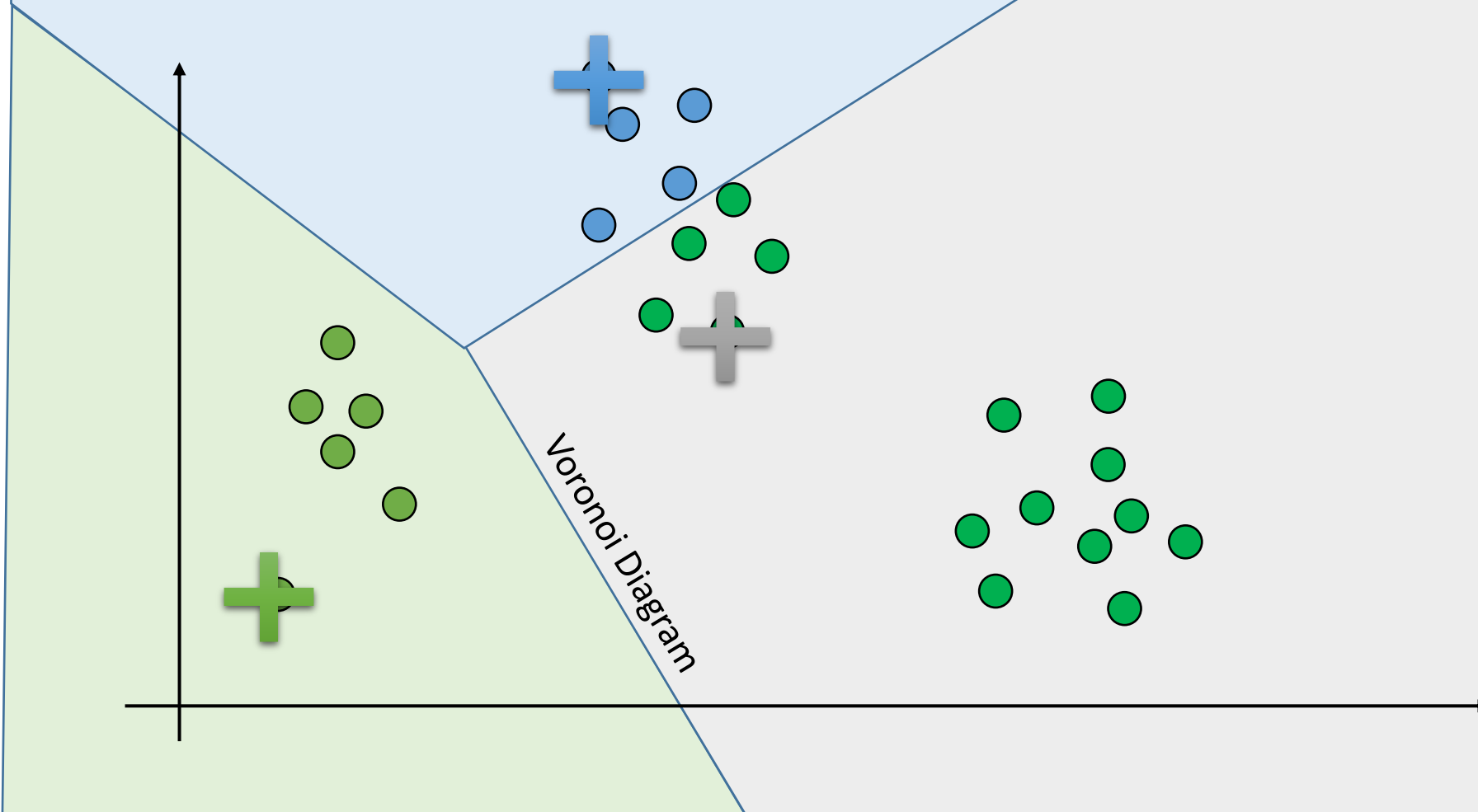
K-Means Clustering: *Intuition*

- For each data point find the cluster nearest center



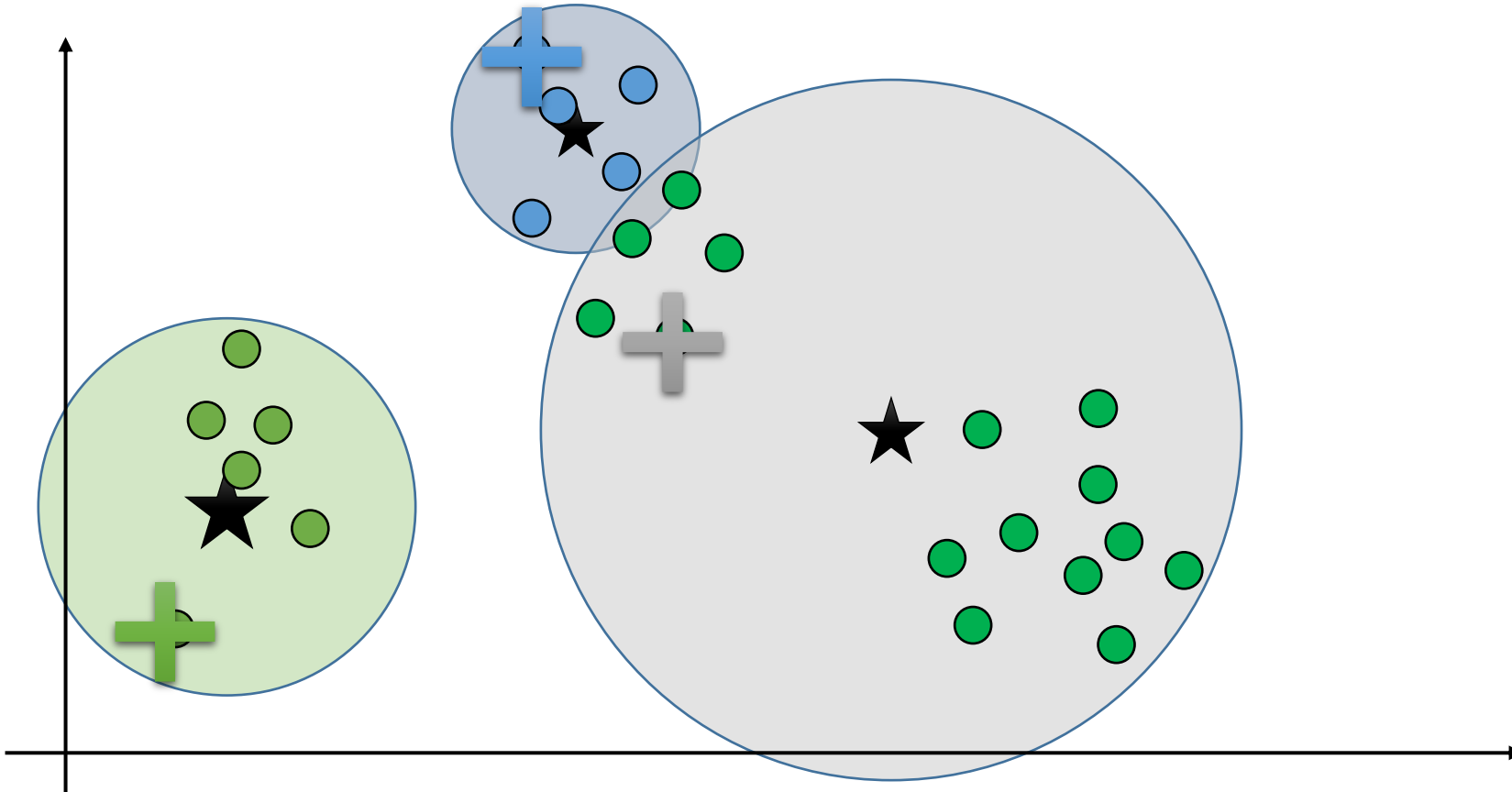
K-Means Clustering: *Intuition*

- For each data point find the cluster nearest center



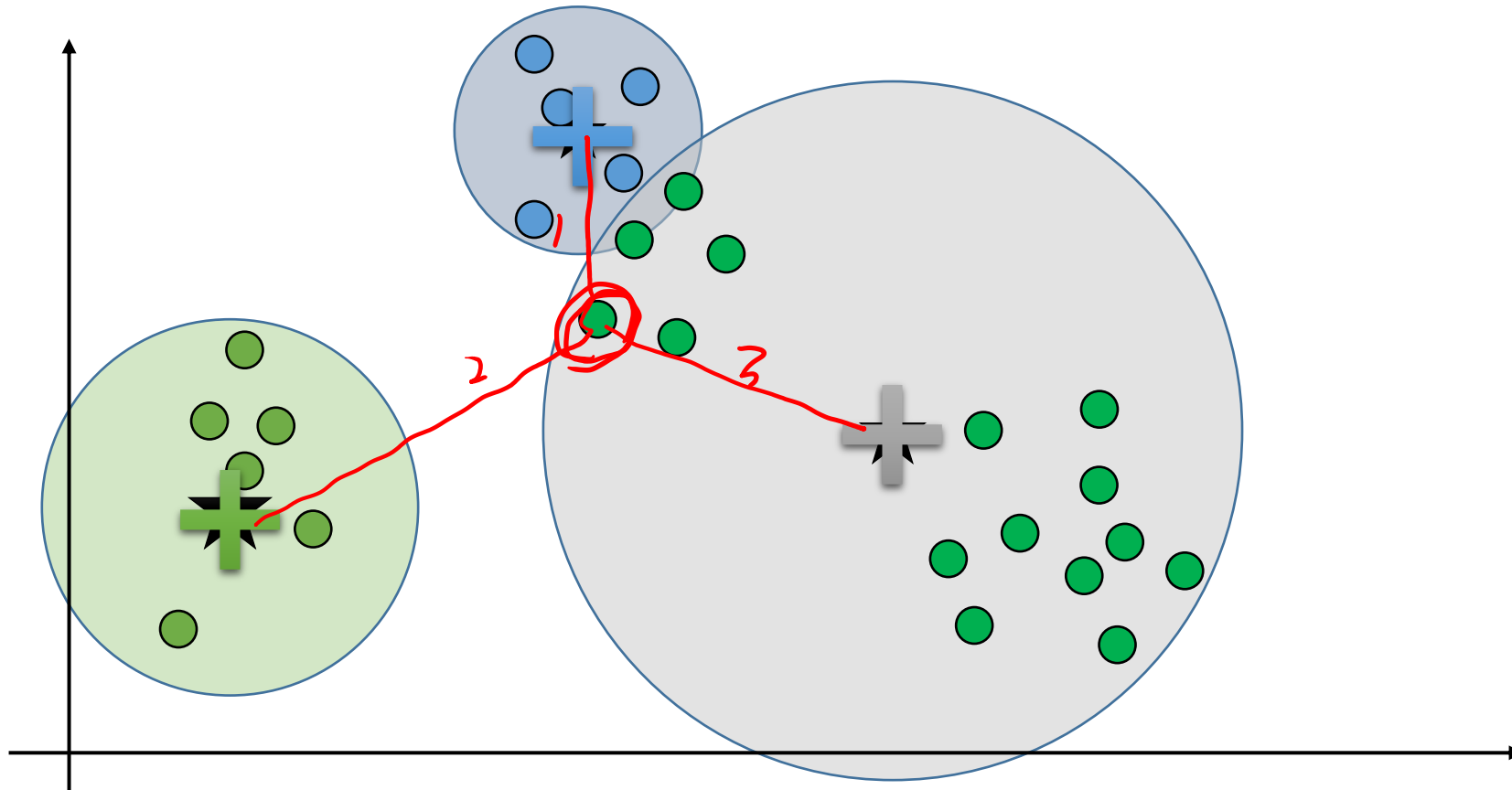
K-Means Clustering: *Intuition*

- Compute mean of points in each “cluster”



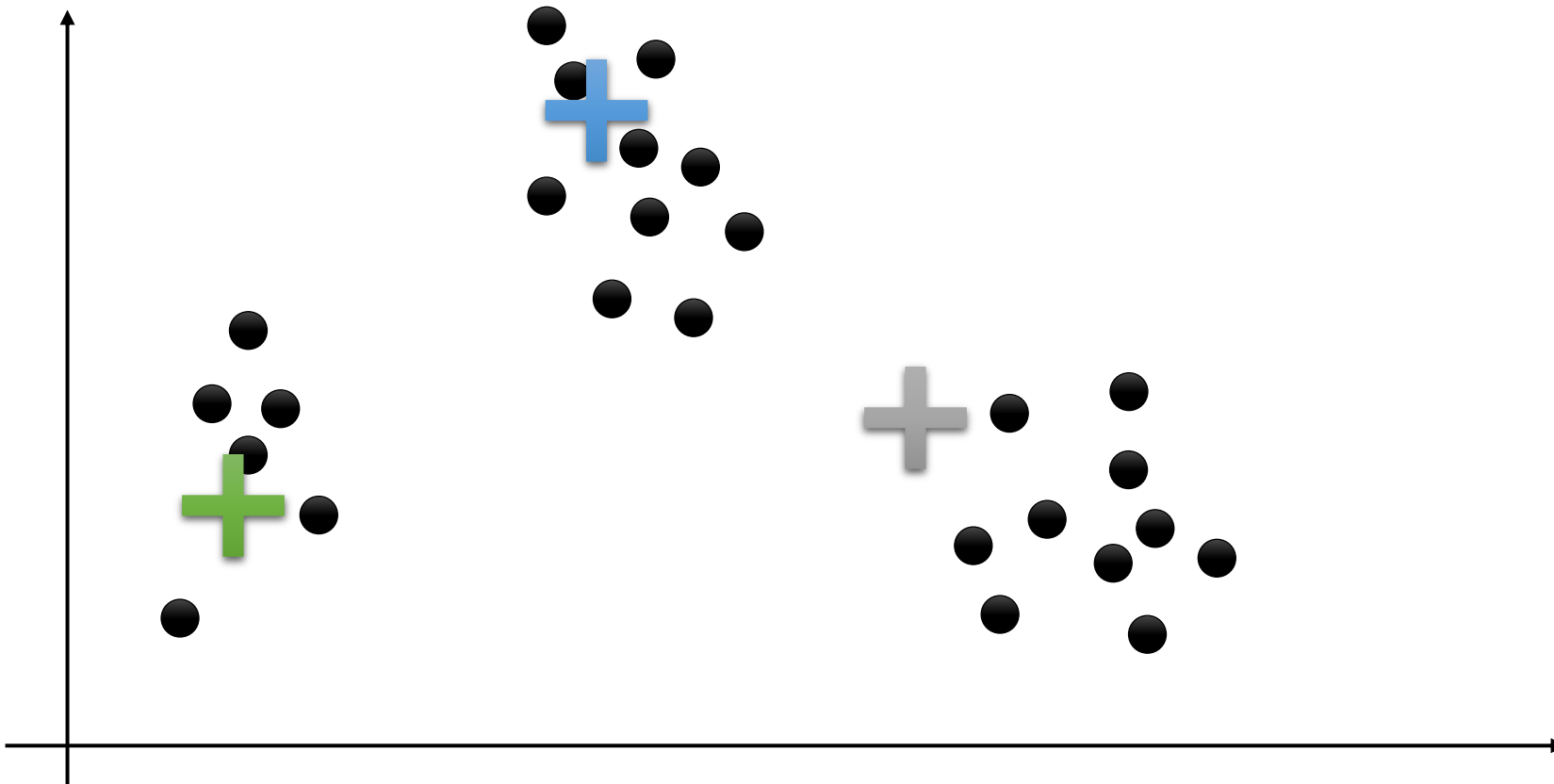
K-Means Clustering: *Intuition*

- Adjust cluster centers to be the mean of the cluster



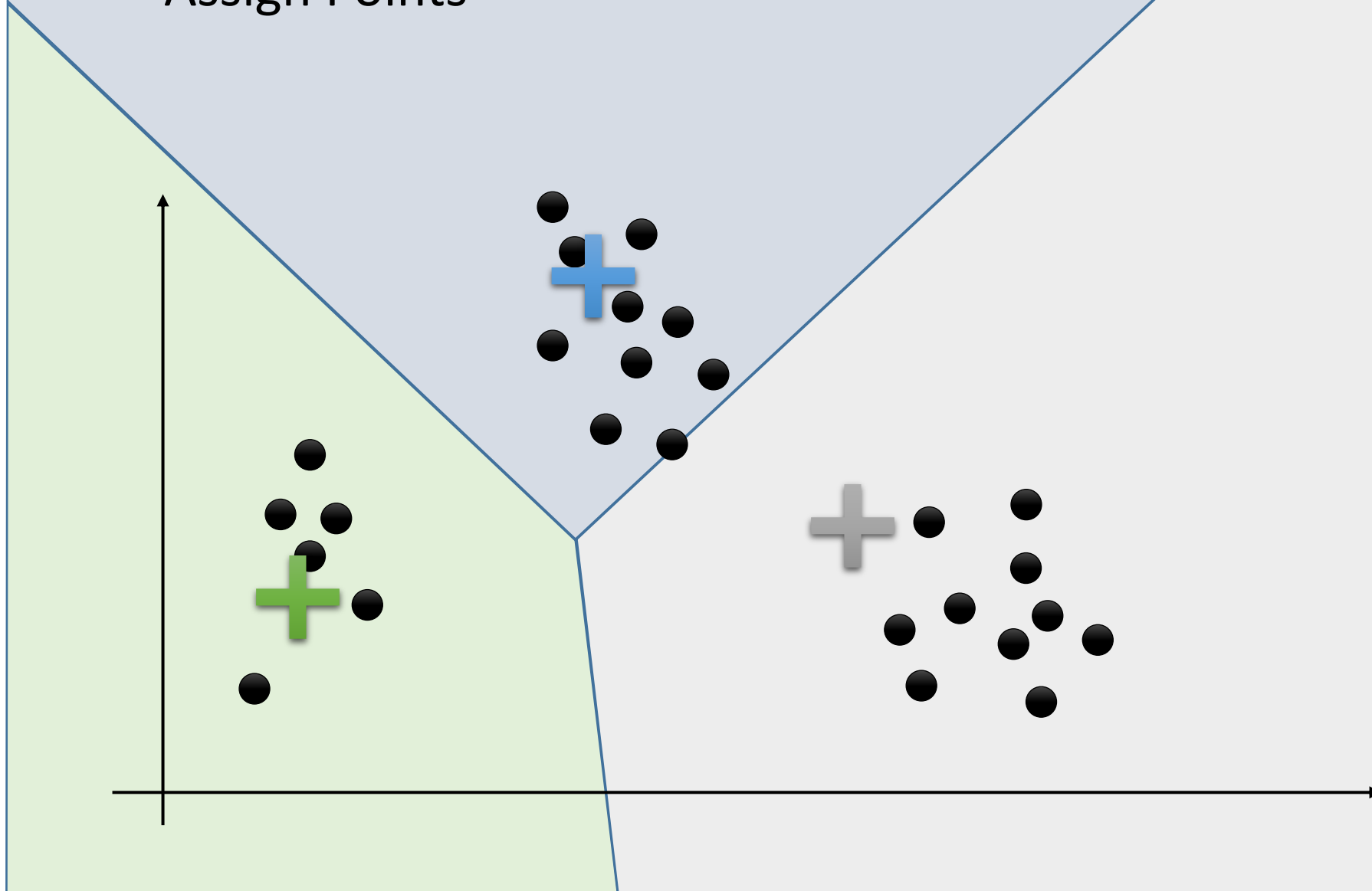
K-Means Clustering: *Intuition*

- Improved?
- Repeat



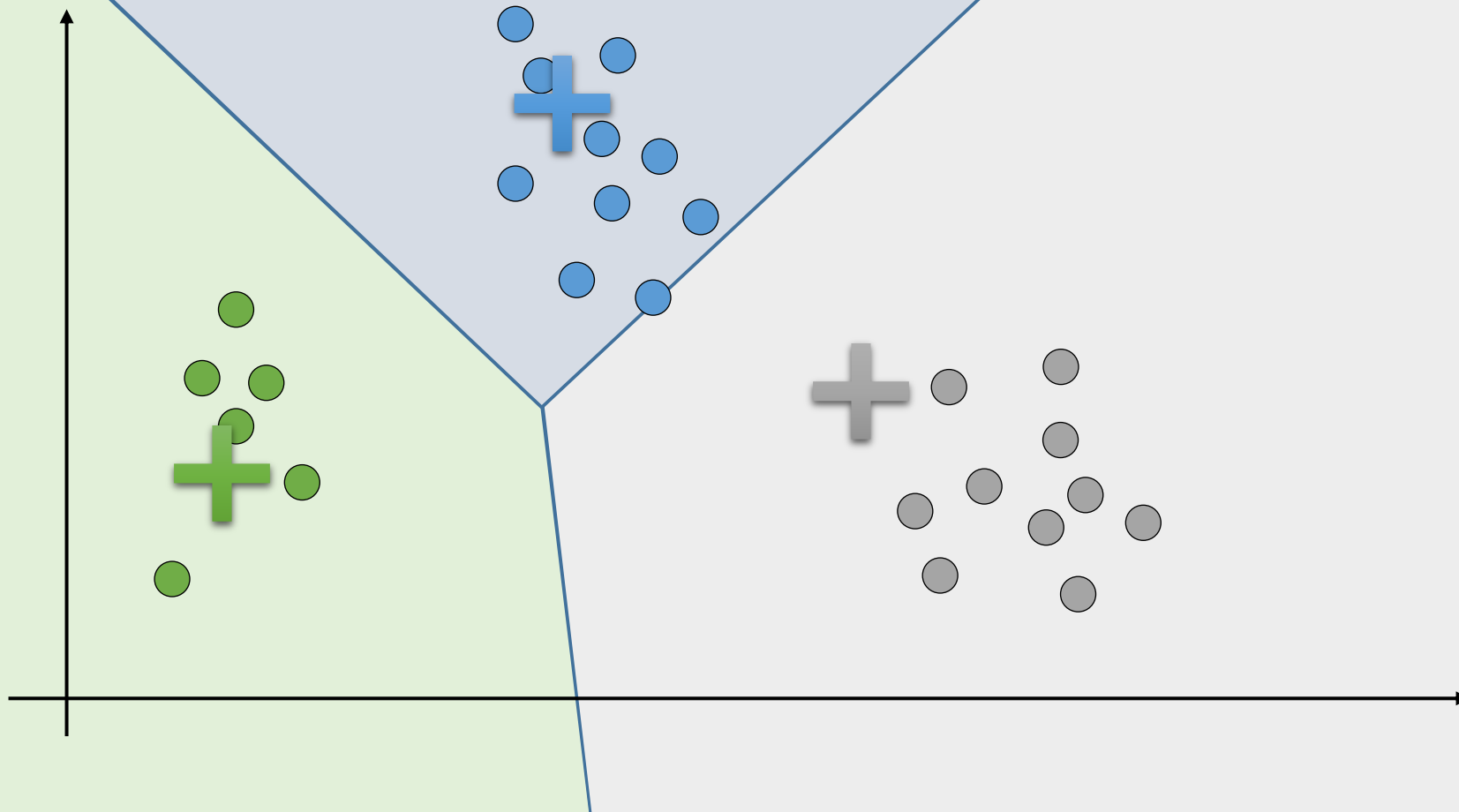
K-Means Clustering: *Intuition*

- Assign Points



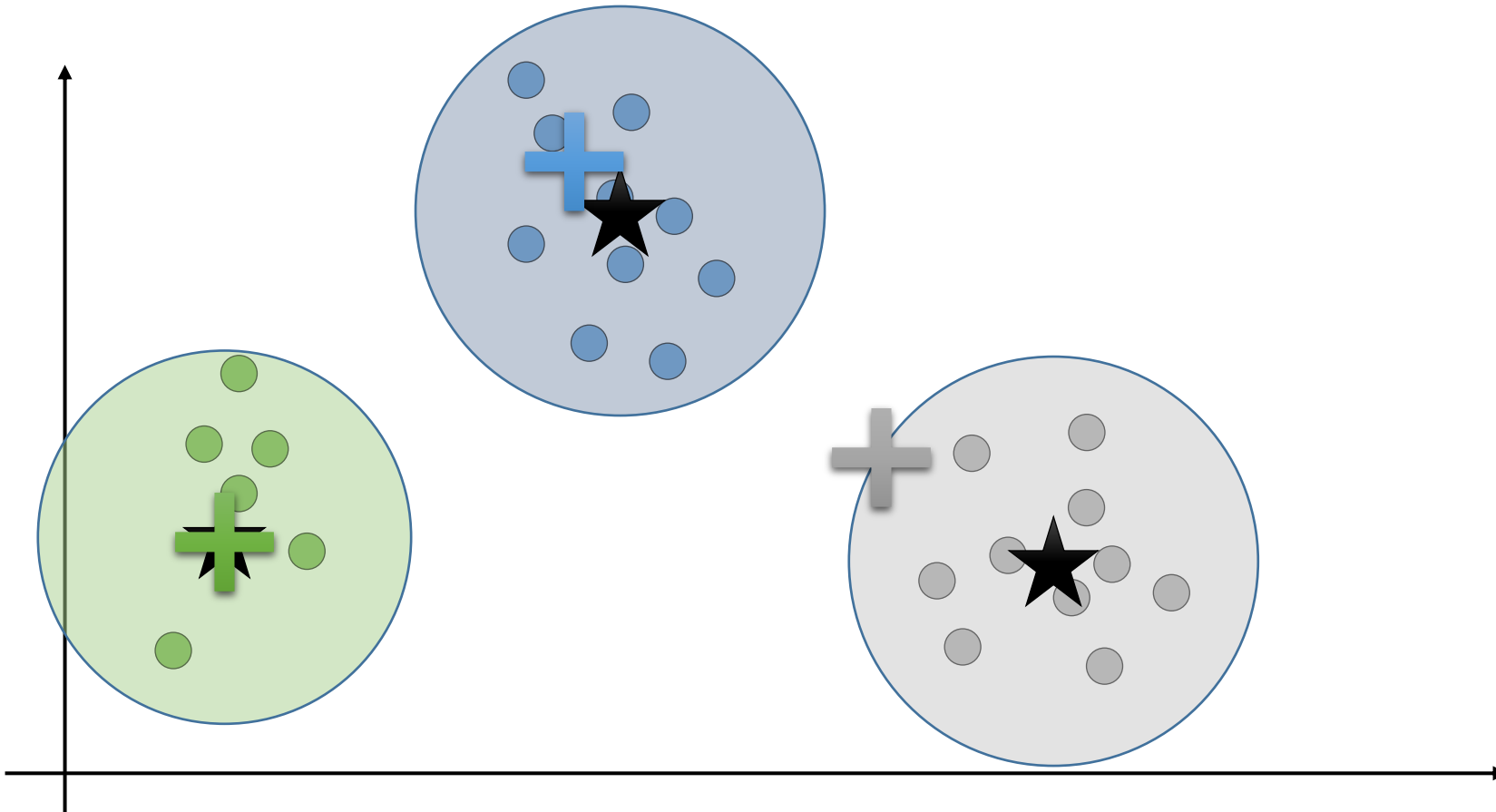
K-Means Clustering: *Intuition*

- Assign Points



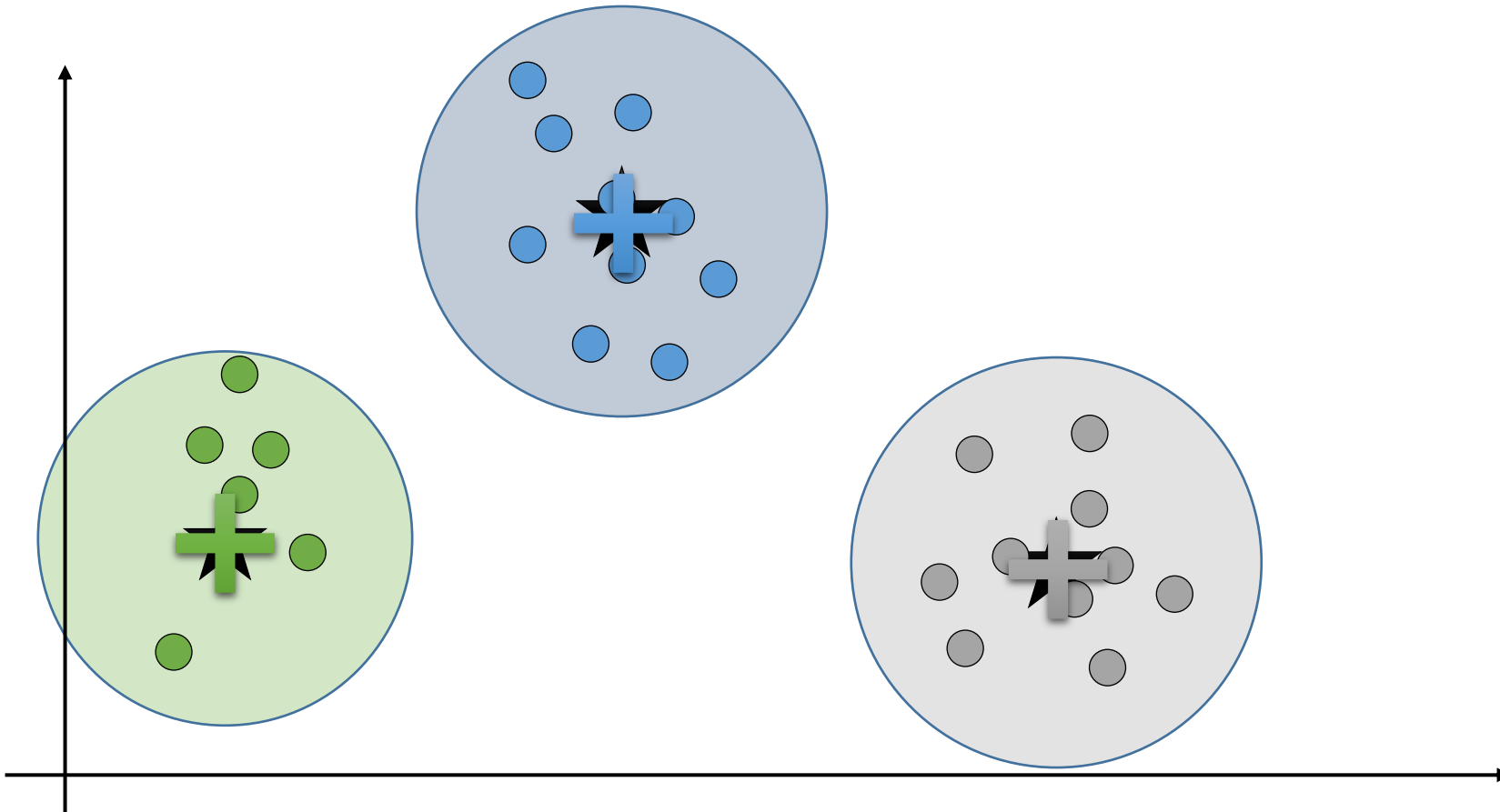
K-Means Clustering: *Intuition*

- Compute cluster means



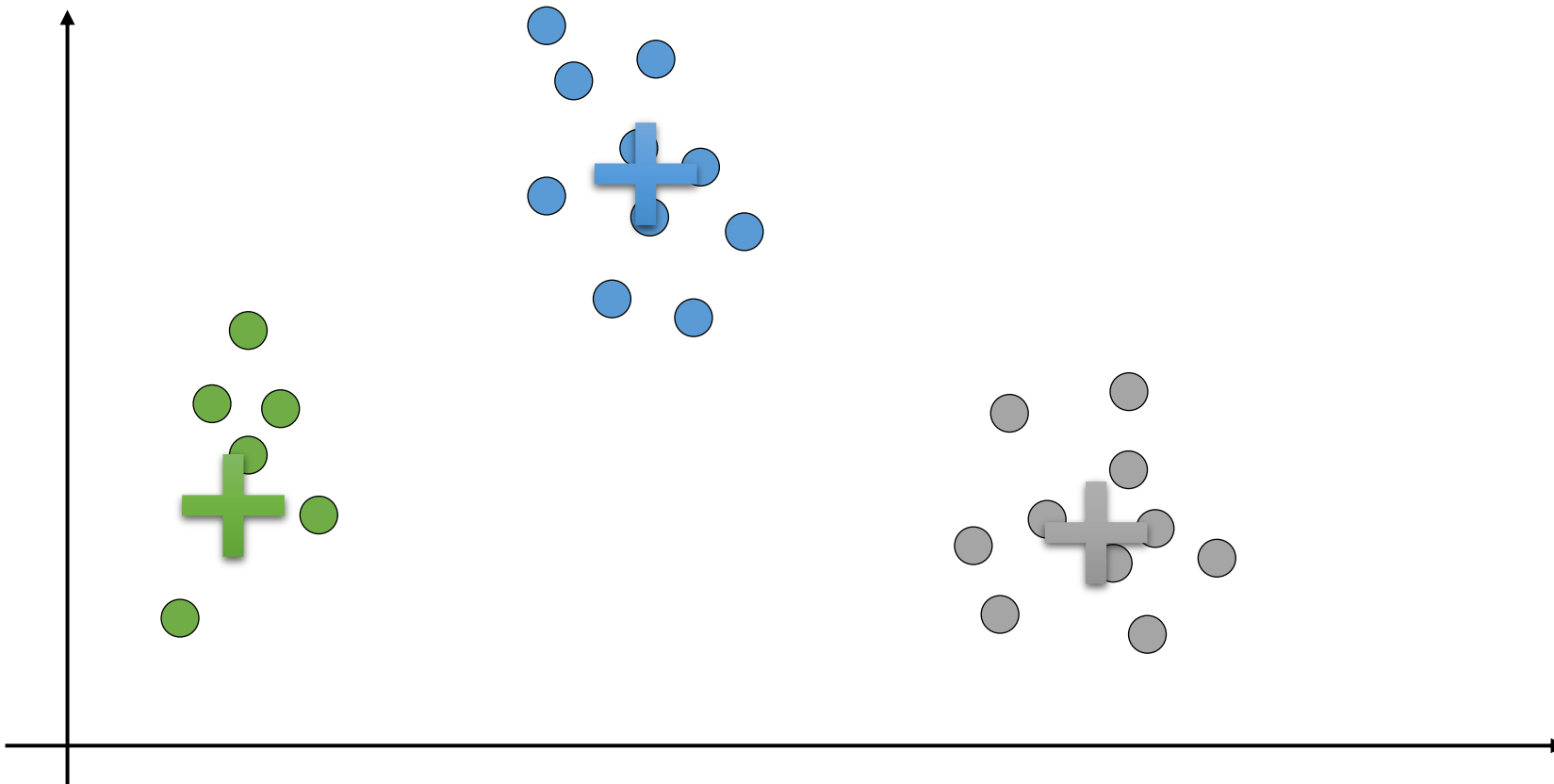
K-Means Clustering: *Intuition*

- Update cluster centers



K-Means Clustering: *Intuition*

- Repeat?
 - Yes to check that nothing changes → Converged!



K-Means Algorithm: Details

`centers ← pick k initial Centers`

`while (centers are changing) {`

`// Compute the assignments (E-Step)`

`asg ← [(x, nearest(centers, x)) for x in data]`

What do we mean by “nearest”:

A: Euclidean Distance

$$\arg \min_{c \in \text{centers}} ||c - x||_2^2 = \sum_{i=1}^d (c_i - x_i)^2$$

K-Means Algorithm: Details

`centers ← pick k initial Centers`

Compute the
“Expected” Assignment

`while` (centers are changing) {

`// Compute the assignments (E-Step)`

`asg ← [(x, nearest(centers, x)) for x in data]`

`// Compute the new centers (M-Step)`

`for` `i` **`in`** `range(k)`:

Find centers that maximize the
data “likelihood”

`centers[i] =`

`mean([x for (x, c) in asg if c == i])`

`}`

K-Means Algorithm: Details

```
centers ← pick k initial Centers
```

```
while (centers are changing) {  
    // Compute the assignments (E-Step)  
    asg ← [(x, nearest(centers, x)) for x in data]  
  
    // Compute the new centers (M-Step)  
    for i in range(k):  
        centers[i] =  
            mean([x for (x, c) in asg if c == i])  
}
```

Guaranteed to
converge!

... to what?

To a local
optimum. 😞

Depends on
Initial Centers

K-Means Algorithm: Details

```
centers ← pick k initial Centers
```

How do we pick initial centers?

```
while (centers are changing) {  
    // Compute the assignments (E-Step)  
    asg ← [(x, nearest(centers, x)) for x in data]  
  
    // Compute the new centers (M-Step)  
    for i in range(k):  
        centers[i] =  
            mean([x for (x, c) in asg if c == i])  
}
```

Guaranteed to
converge!

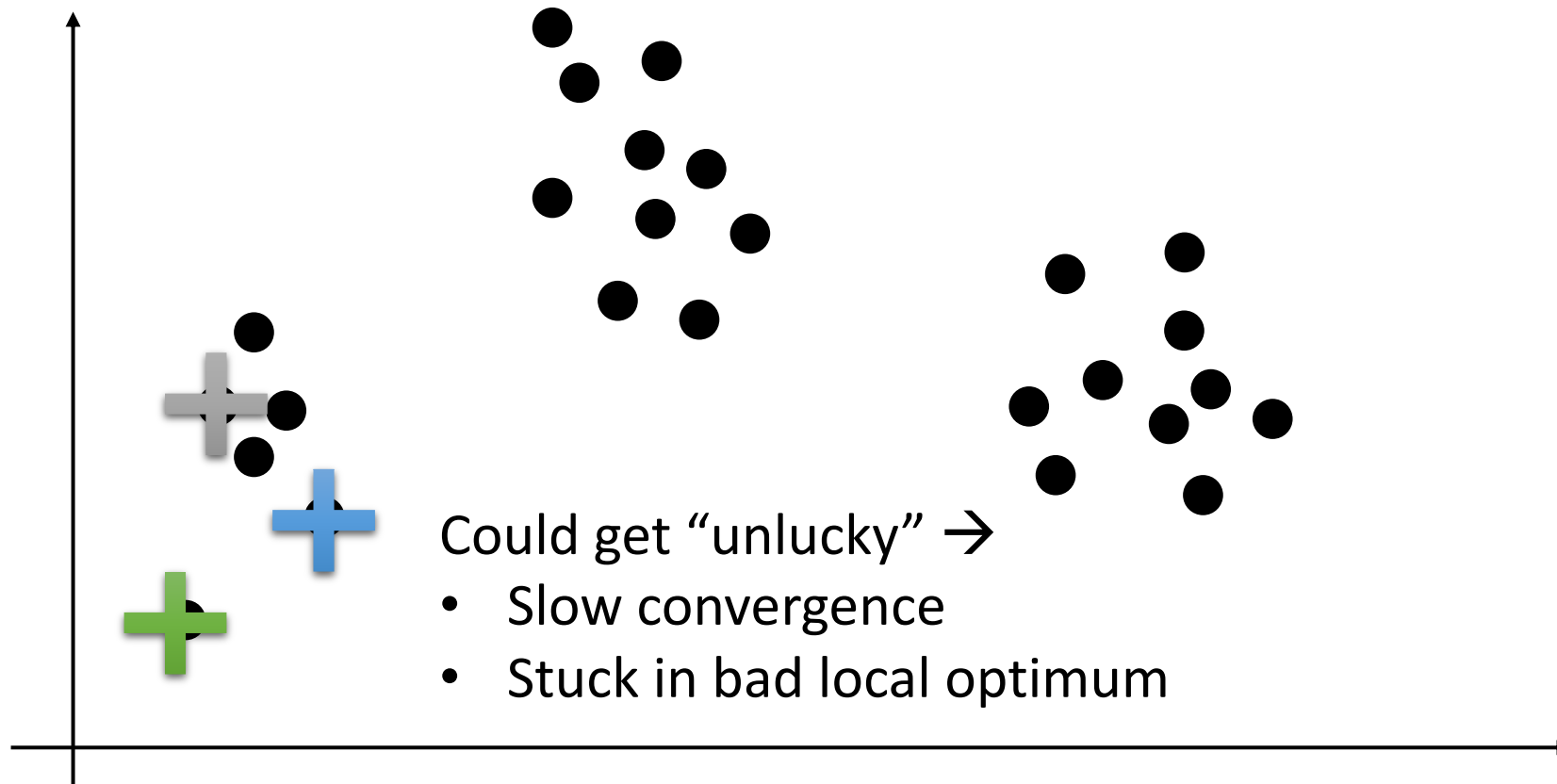
... to what?

To a local
optimum. ☹️

Depends on
Initial Centers

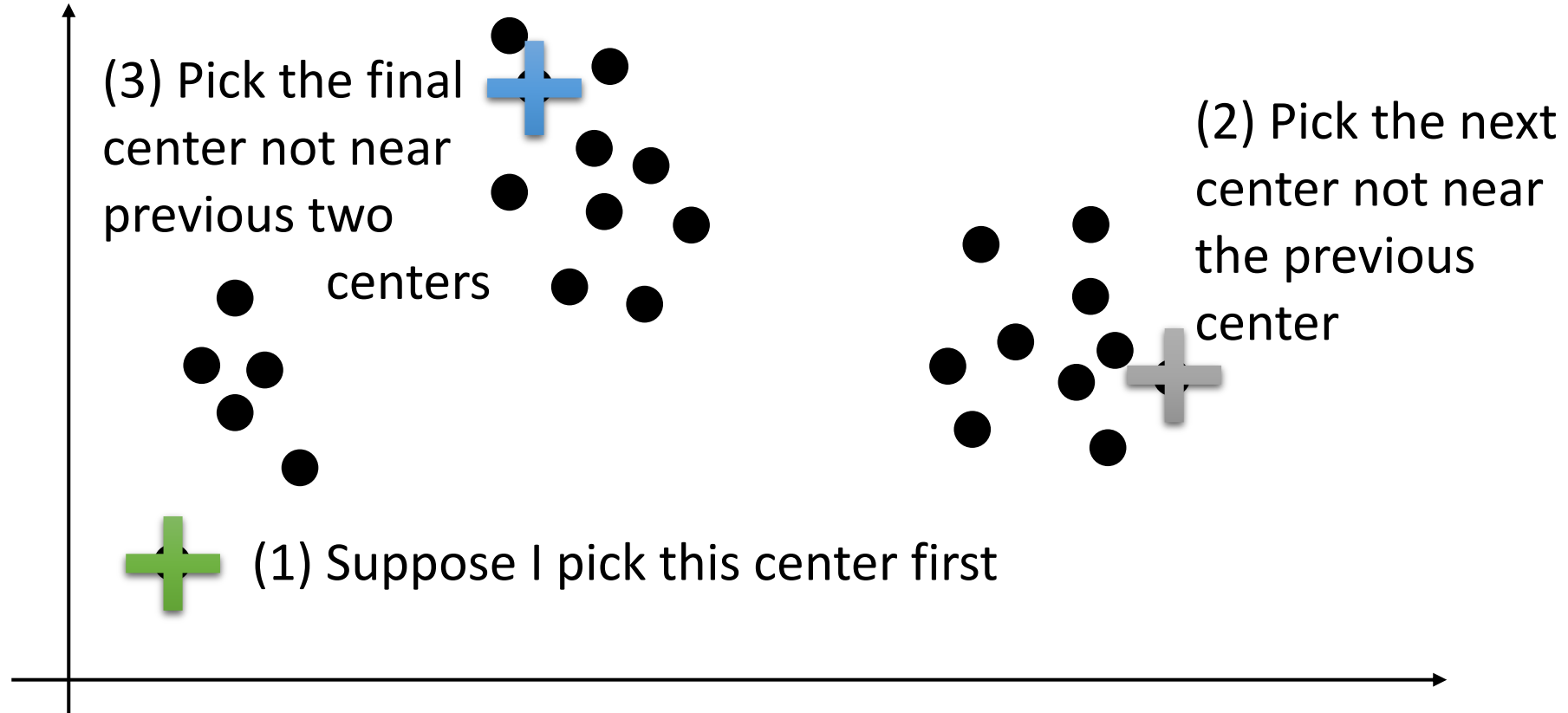
Picking the Initial Centers

- **Simple Strategy:** select k points at random
 - What could go wrong?



Picking the Initial Centers

- **Better Strategy:** kmeans++
 - Randomized approx. algorithm
 - Intuition select points that are not near existing centers



K-Means++ Algorithm

```
centers ← set(randomly select a single point)
```

```
while len(centers) < k:
```

```
    # Compute the distance of each point
```

```
    # to its nearest center  $dSq = d^2$ 
```

```
     $dSq \leftarrow [(x, \text{dist\_to\_nearest}(\text{centers}, x)^2) \text{ for } x \text{ in data}]$ 
```

```
    # Sample a new point with probability
```

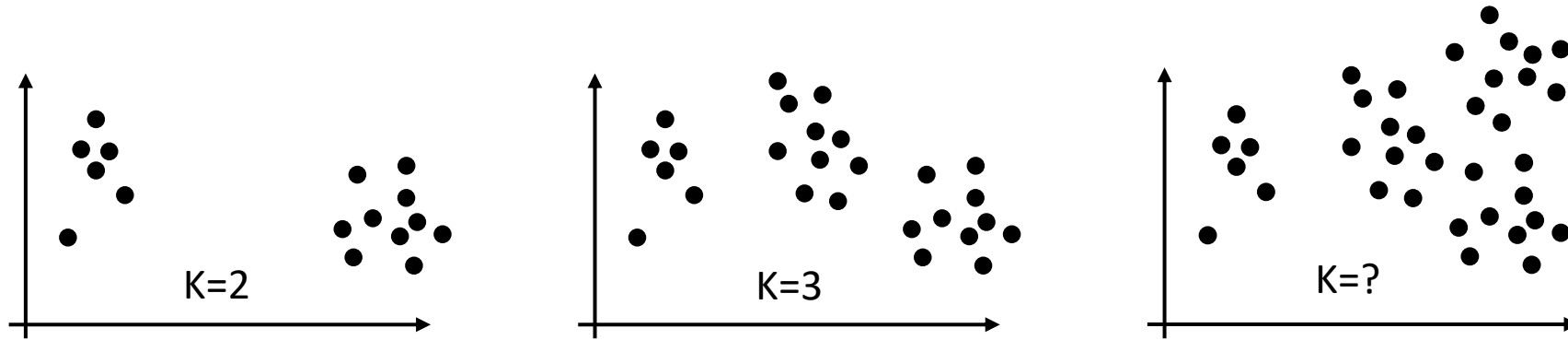
```
    # proportional to  $dSq$ 
```

```
     $c \leftarrow \text{sample\_one}(\text{data}, \text{prob} = dSq / \text{sum}(dSq))$ 
```

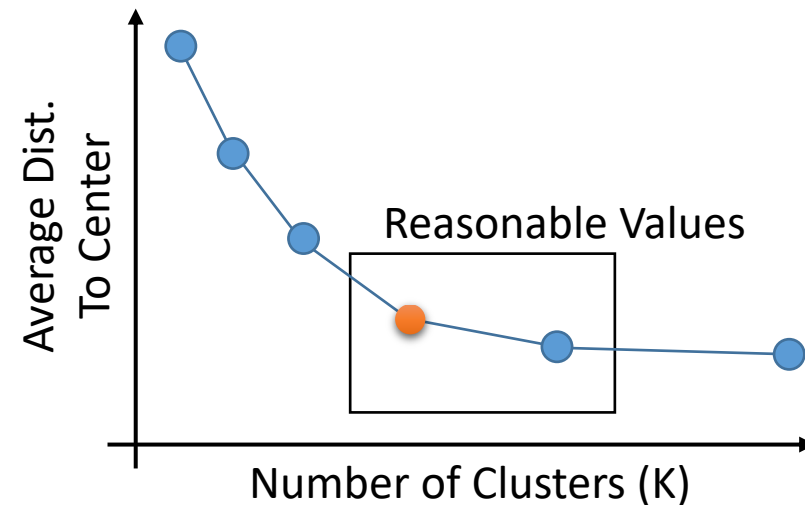
```
    # Update the clusters
```

```
    centers.add(c)
```


How do we choose K?



- Basic Elbow Method (Easy)
 - Try range of K-values and plot average distance to centers
- Cross-Validation (Better)
 - Repeatedly split the data into training and validation datasets
 - Cluster the training dataset
 - Measure Avg. Dist. To Centers on validation data

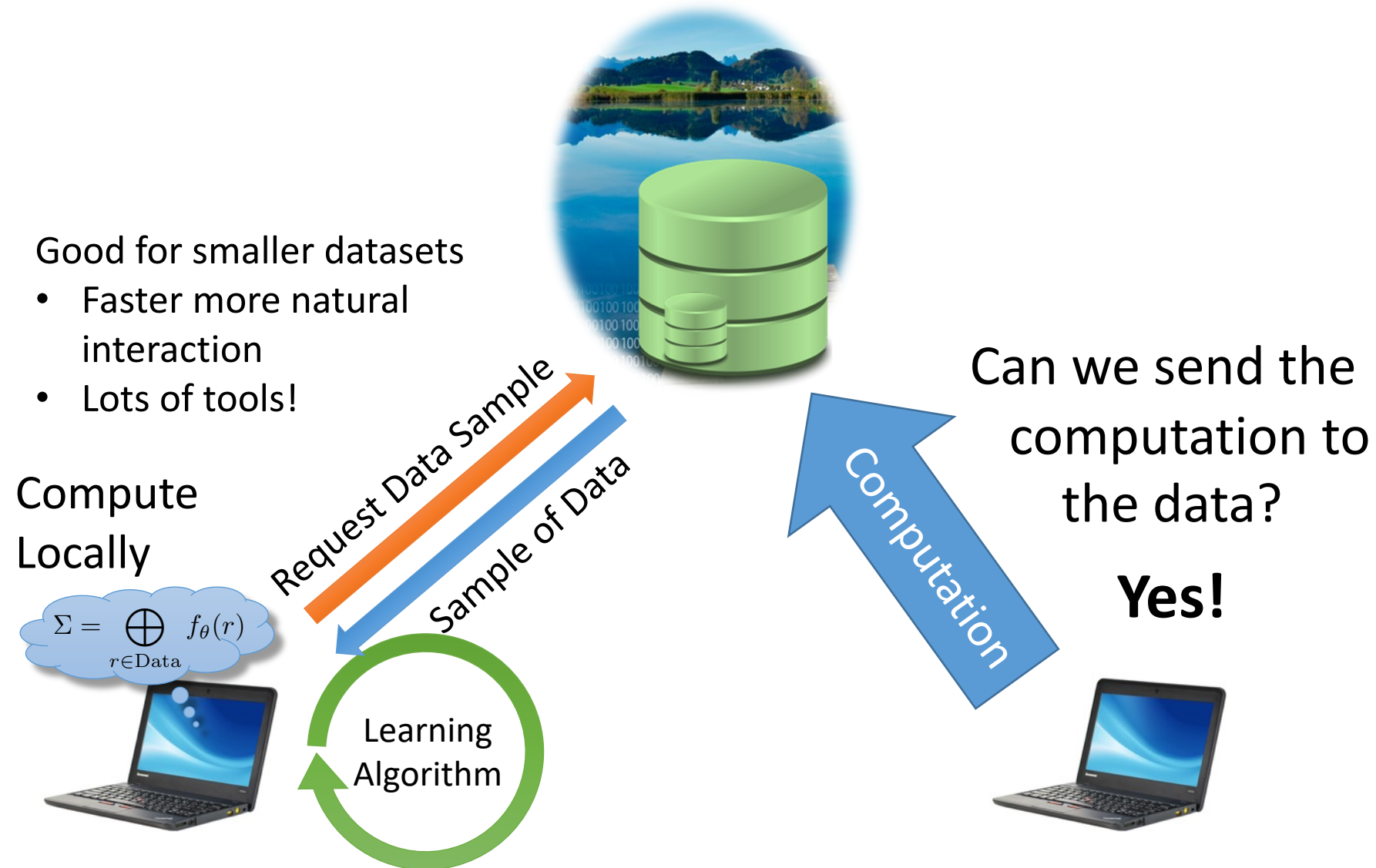


K-Means +



How do we run k-means on the
data warehouse / data lake?

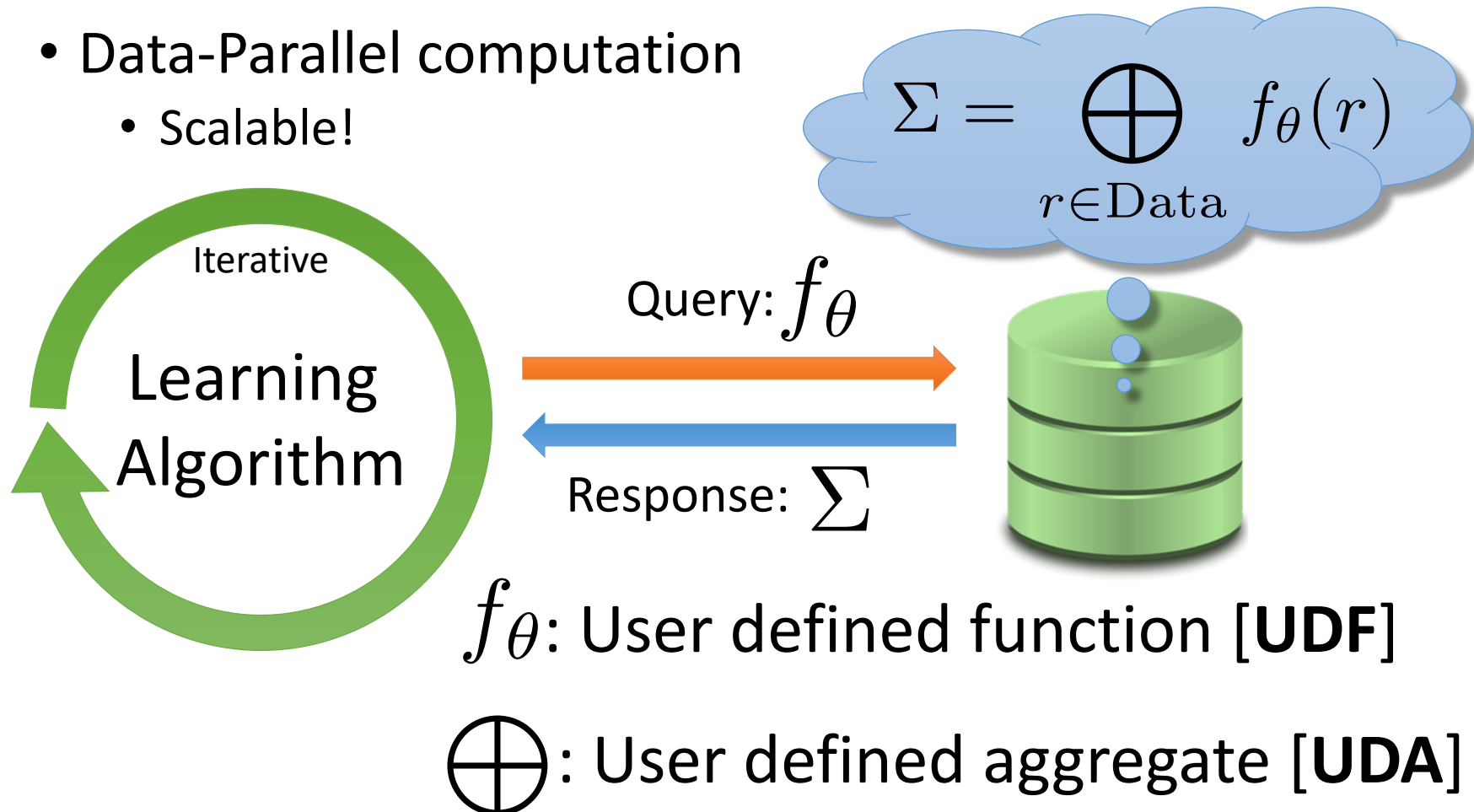
Interacting With the Data



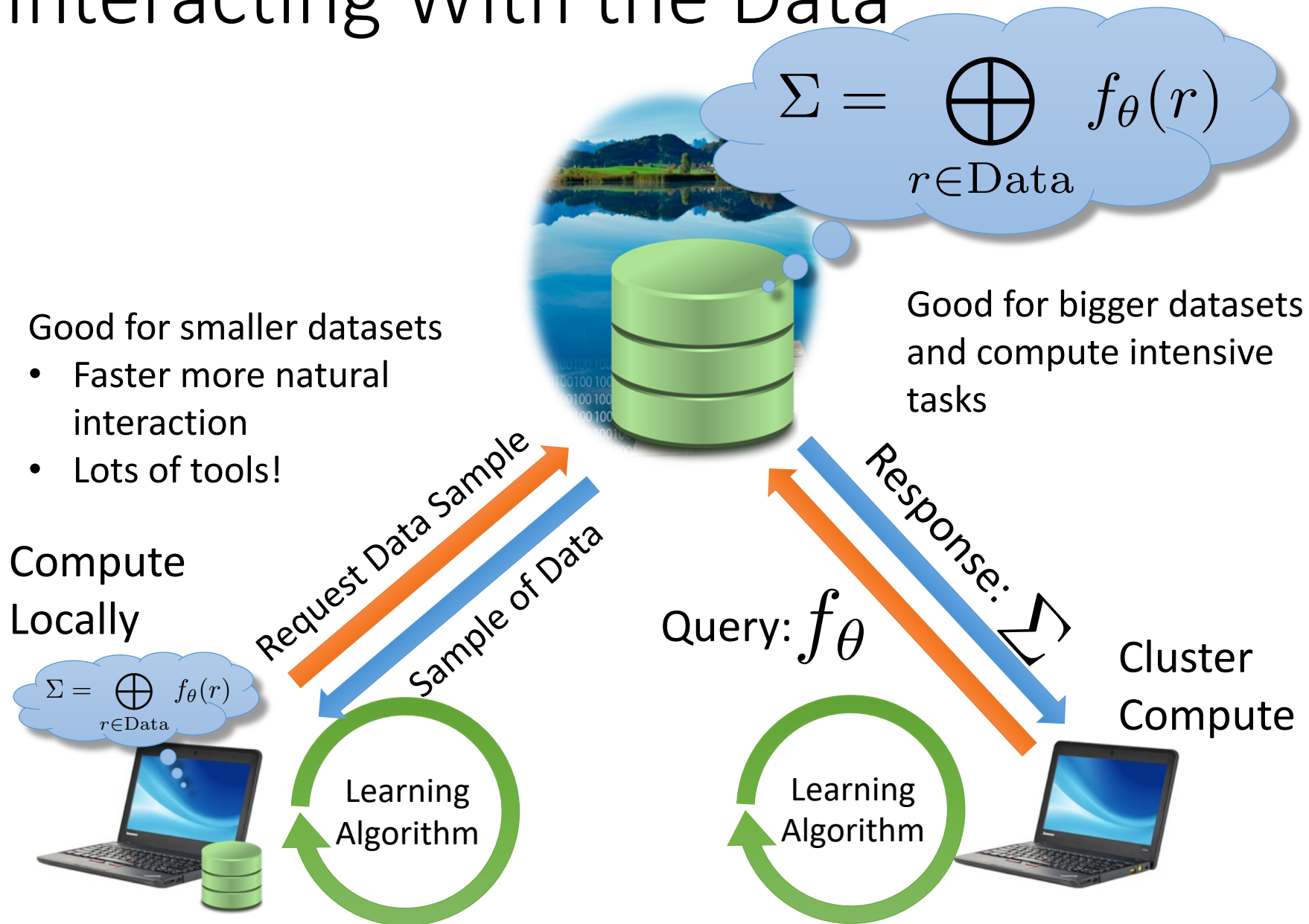
Statistical Query Pattern

Common Machine Learning Pattern

- Computing aggregates of user defined functions
- Data-Parallel computation
 - Scalable!



Interacting With the Data



Can we express K-Means in the Statistical Query Pattern?

```
centers ← pick k initial Centers
while (centers are changing):
    // Compute the assignments (E-Step)
    asg ← [(x, nearest(centers, x)) for x in data]
    for i in range(k): // Compute the new centers (M-Step)
        centers[i] = mean([x for (x, c) in asg if c == i])
```

Query returns all
the data ...

Merge with M-Step
→
Statistical Query
Pattern

```
centers ← pick k initial Centers
while (centers are changing):
    for i in range(k):
        new_centers[i] =
            mean([x for x in data if nearest(centers, x) == i])
    centers = new_centers
```

Can we express K-Means in the Statistical Query Pattern?

```
centers ← pick k initial Centers
while (centers are changing):
    for i in range(k):
        new_centers[i] =
            mean([x for x in data if nearest(centers, x) == i])
    centers = new_centers
```

Group by query:

```
SELECT nearest_UDF(centers, x) AS cid, mean_UDA(x)
FROM data GROUPBY cid
```

K-Means in Map-Reduce

- **MapFunction**(*old_centers*, *x*)
 - Compute the index of the nearest old center
 - Return (**key** = *nearest_centers*, **value** = (*x*, 1))
- **ReduceFunction** combines values and counts
 - For each cluster center (Group By)
- Data system returns aggregate statistics:

$$s_i = \sum_{x \in \text{Cluster } i} x_i \quad \text{and} \quad n_i = \sum_{x \in \text{Cluster } i} 1$$

- ML algorithm computes new centers: $\mu_i = s_i / n_i$

Can we express K-Means++ in the Statistical Query Pattern?

- *Yes*, however there is a better version: **K-Means||**
 - More complex but much faster
- Or you can parallelize **K-Means++** directly
 - Requires more passes
- Challenging Step?
 - Parallel weighted sampling:

```
sample_one(data, prob = dSq / sum(dSq))
```

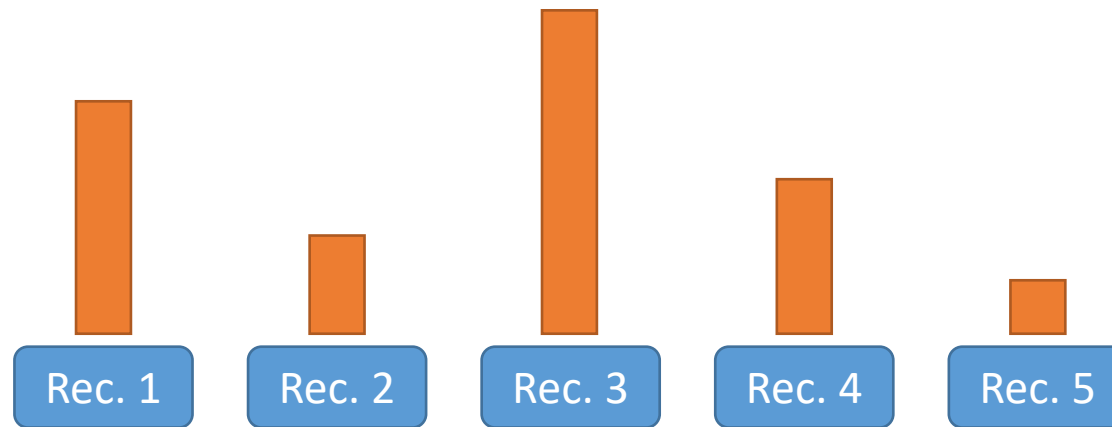
- How do you select one point uniformly at random?

Res-A: weighted reservoir sampling

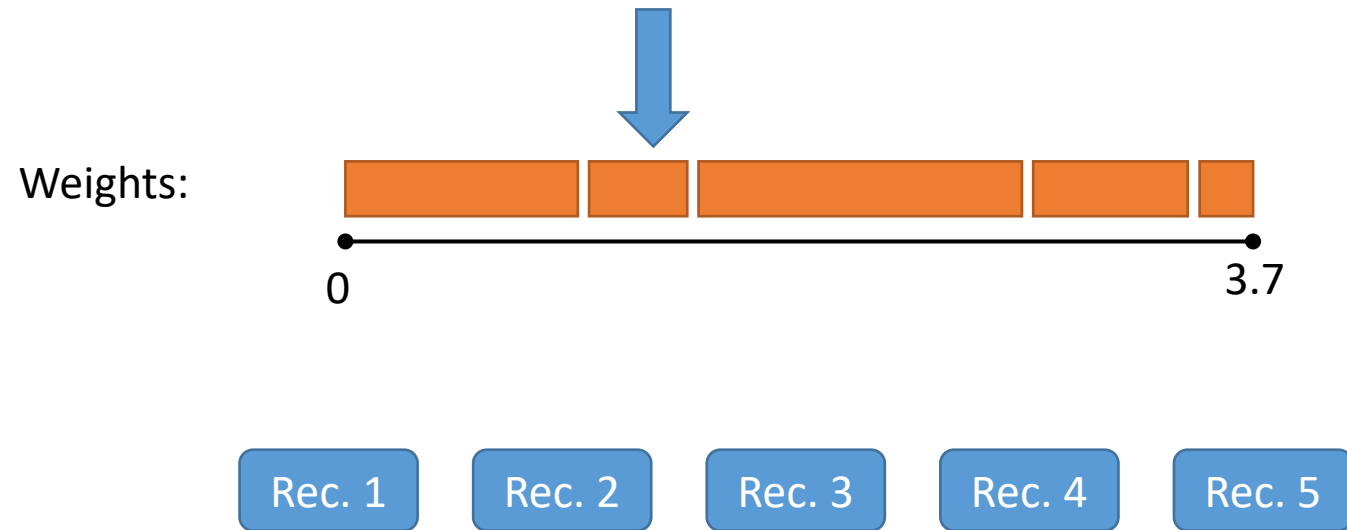
- **Goal:** Sample k records from a stream where record i is included in the sample with probability *proportional to w_i*

How would we normally sample k records?

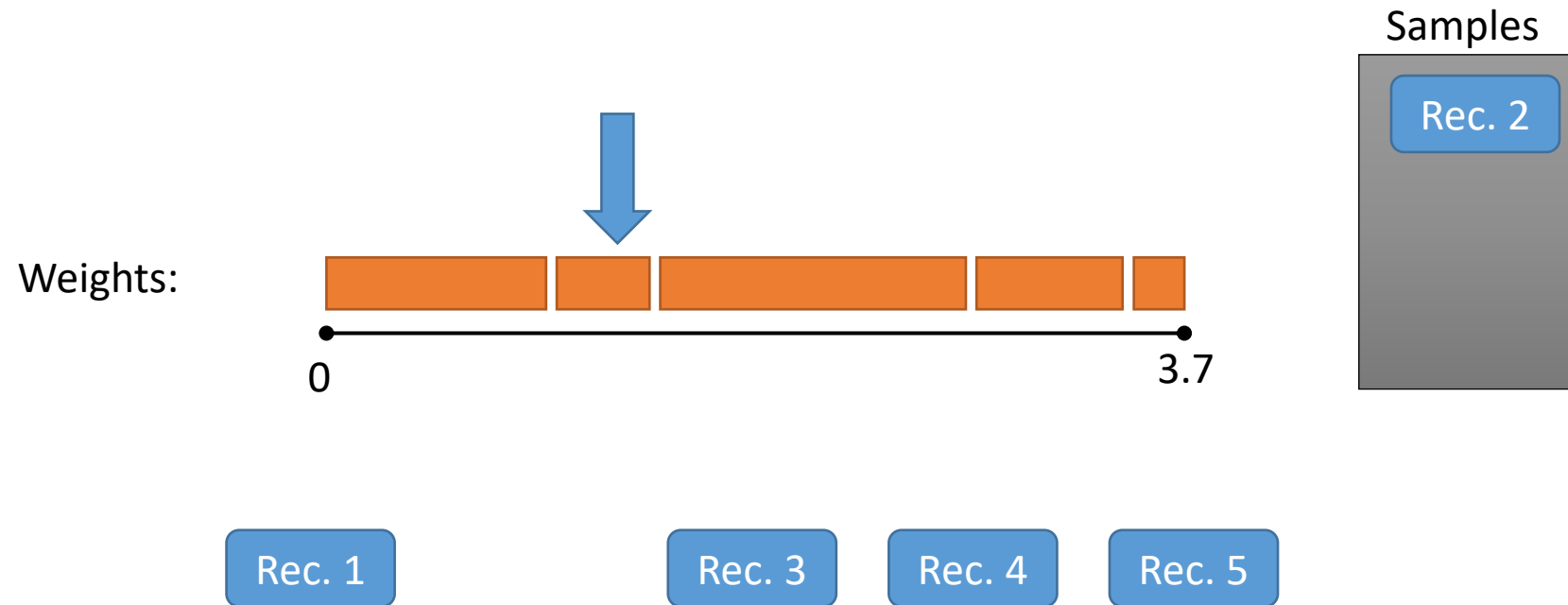
Weights:



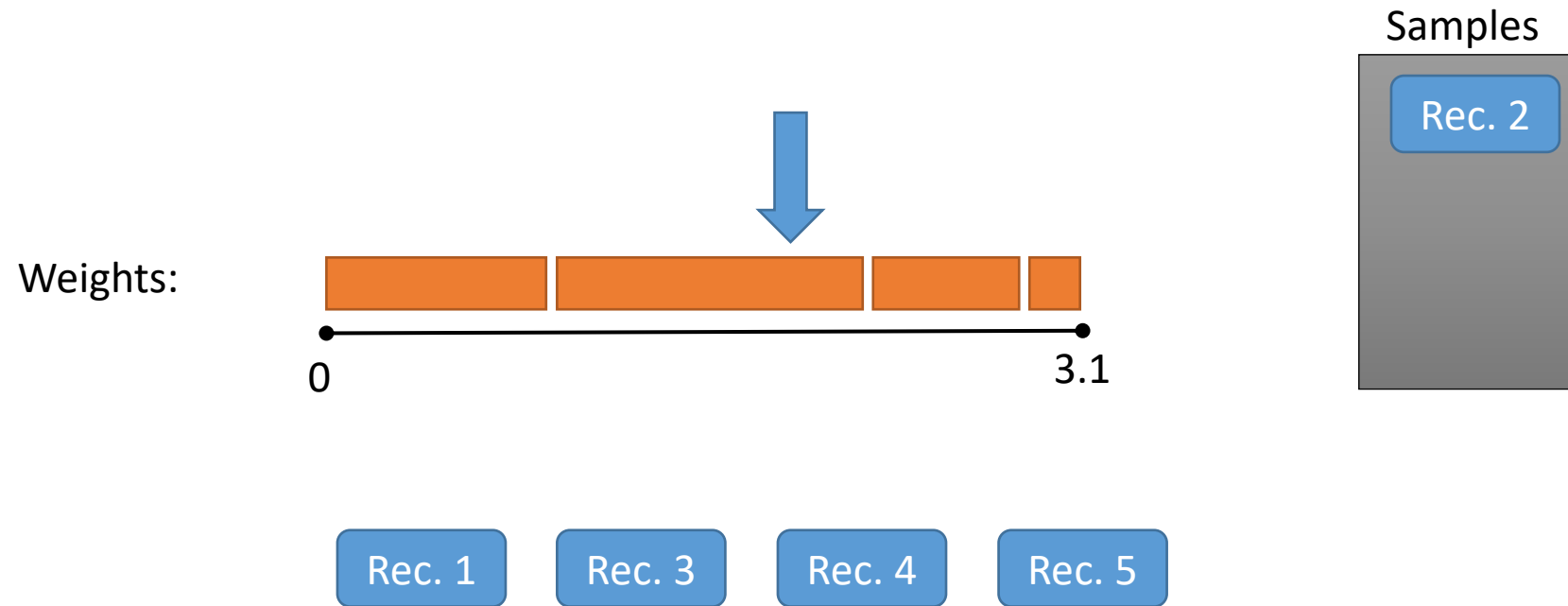
Draw a random number uniformly between **0** and **3.7**



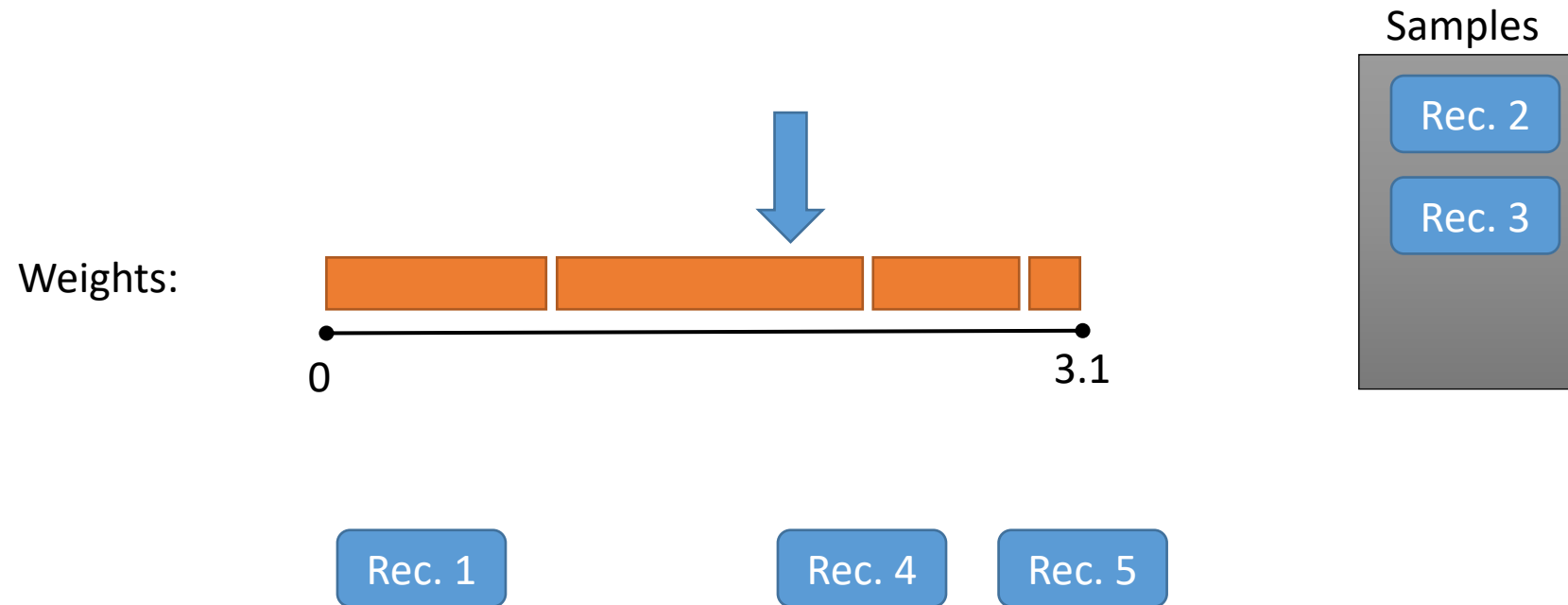
Sample the corresponding record and remove the weight.



Draw a random number uniformly between **0** and **3.1**



We want to do this in **one pass** *without* ever knowing the **sum** of the weights!



Res-A: weighted reservoir sampling

- **Goal:** Sample k records from a stream where record i is included in the sample with probability proportional to w_i

- **Algorithm:**

- For each record i draw a uniform random number:

$$u_i \sim \text{Unif}(0, 1)$$

- Select the top- k records ordered by: u_i^{1/w_i}

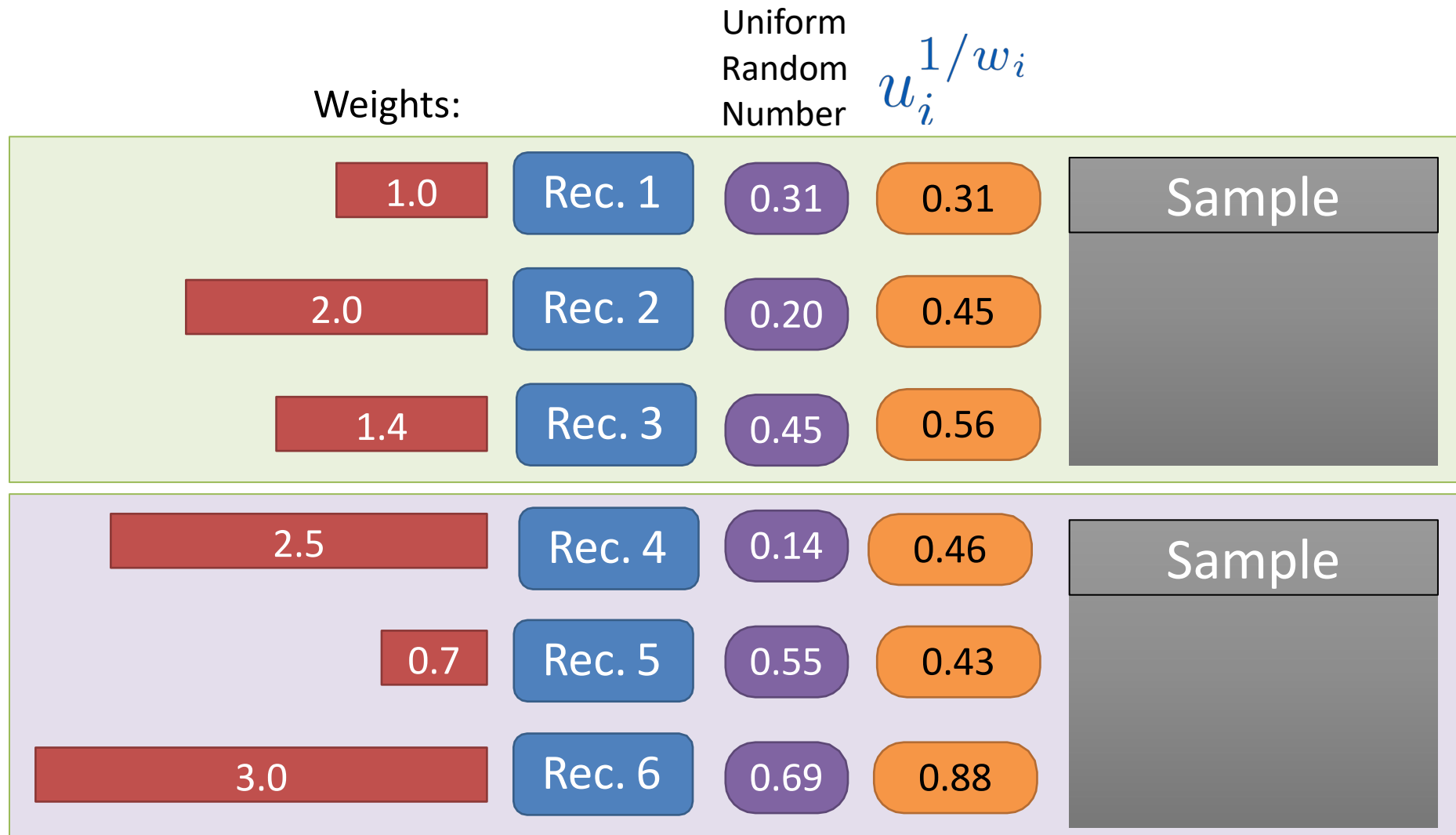
- **Common ML Pattern?**

- **Query Function:** $[pow(rand(), 1 / record.w), record]$
 - **Agg. Function:** *top-k heap*

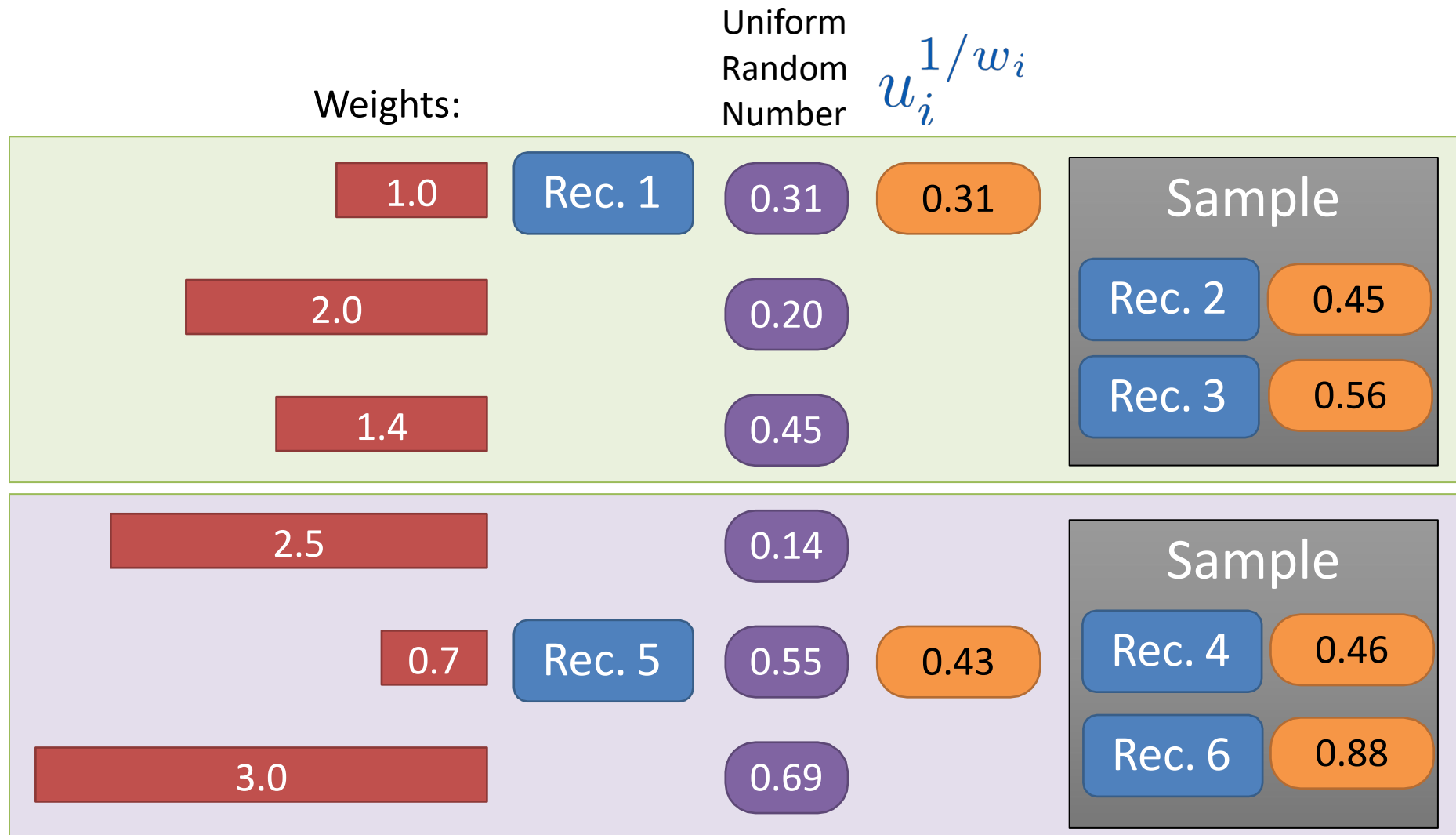
Illustrating Res-A Algorithm



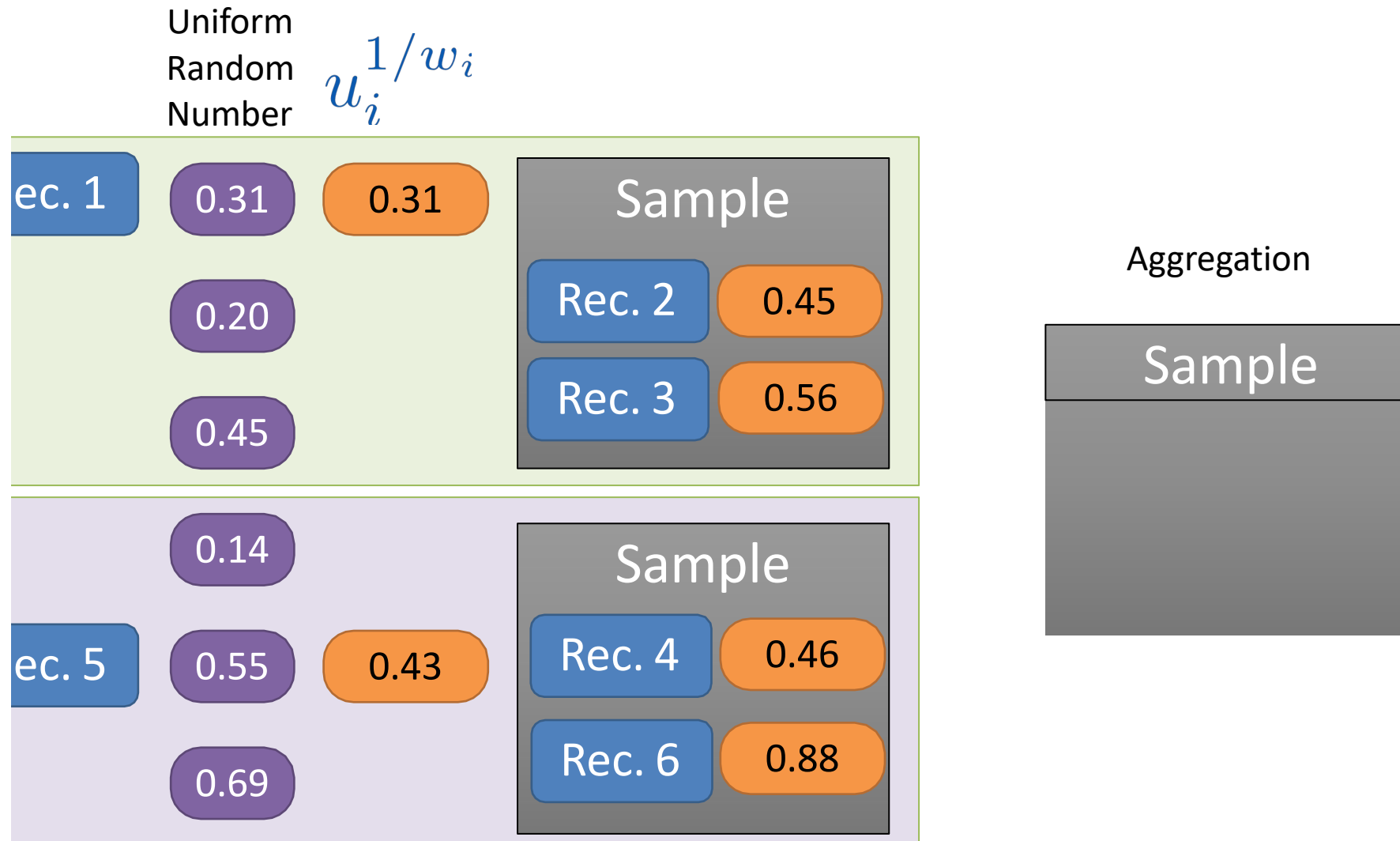
Parallelizing Res-A Algorithm



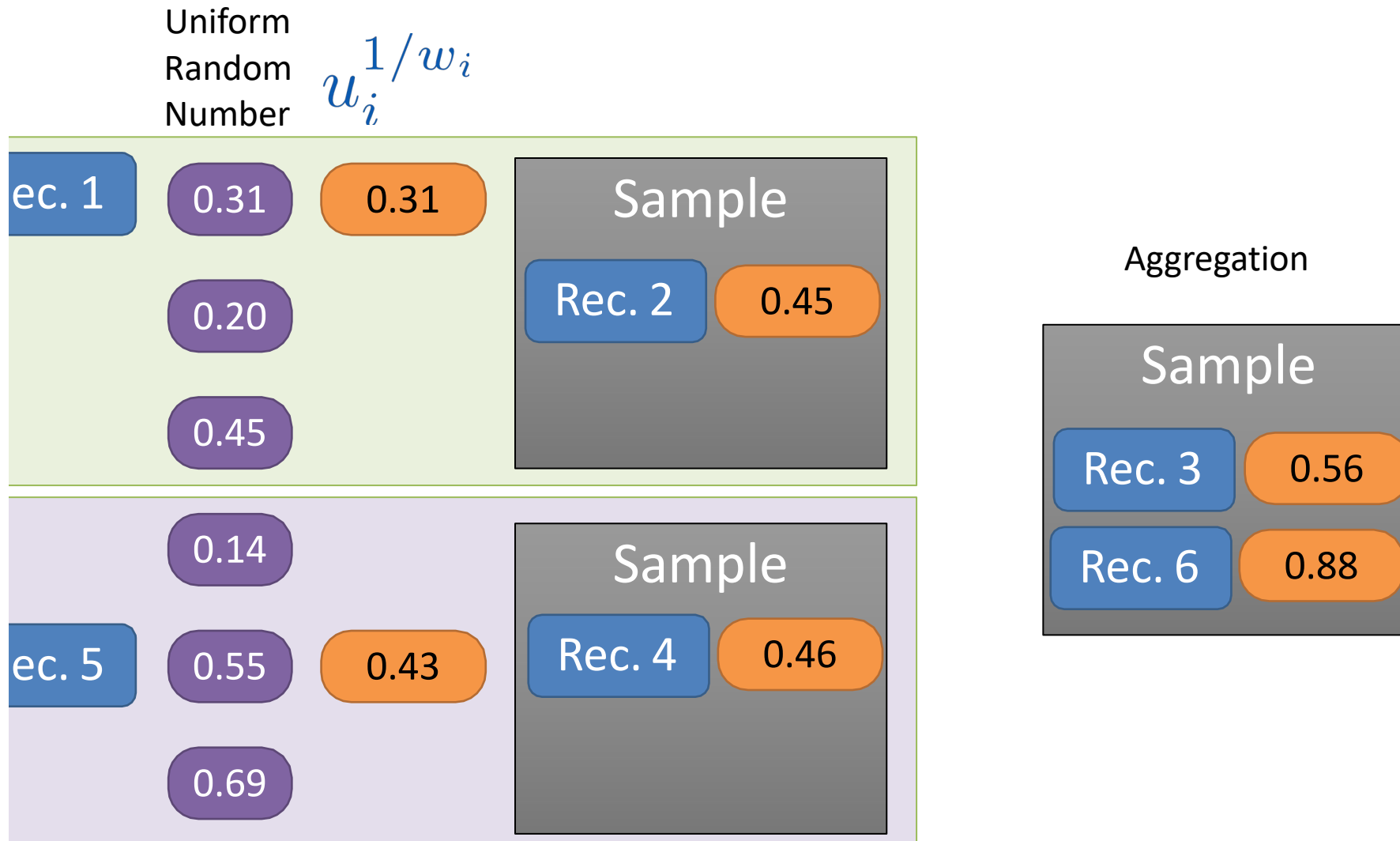
Parallelizing Res-A Algorithm



Parallelizing Res-A Algorithm



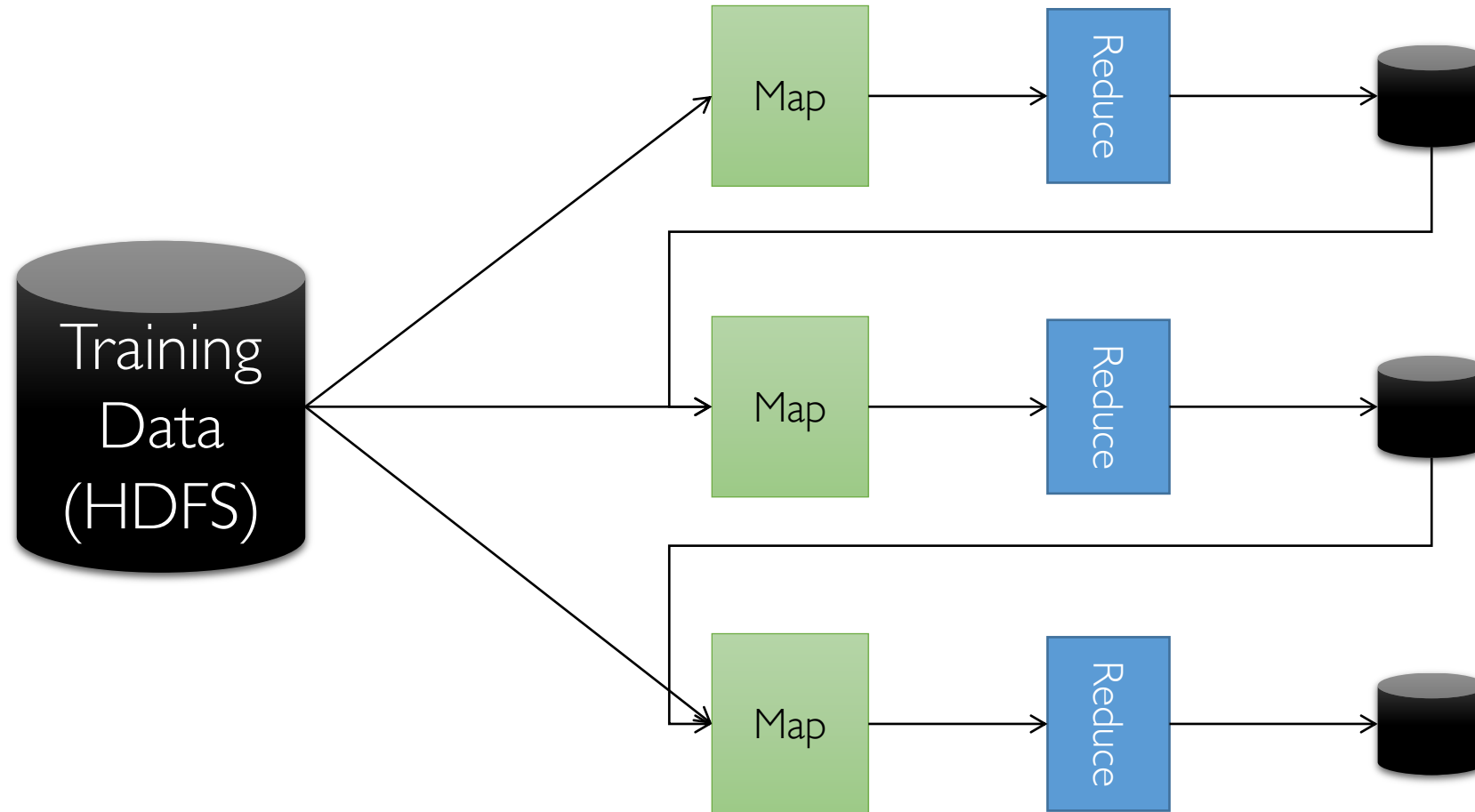
Parallelizing Res-A Algorithm



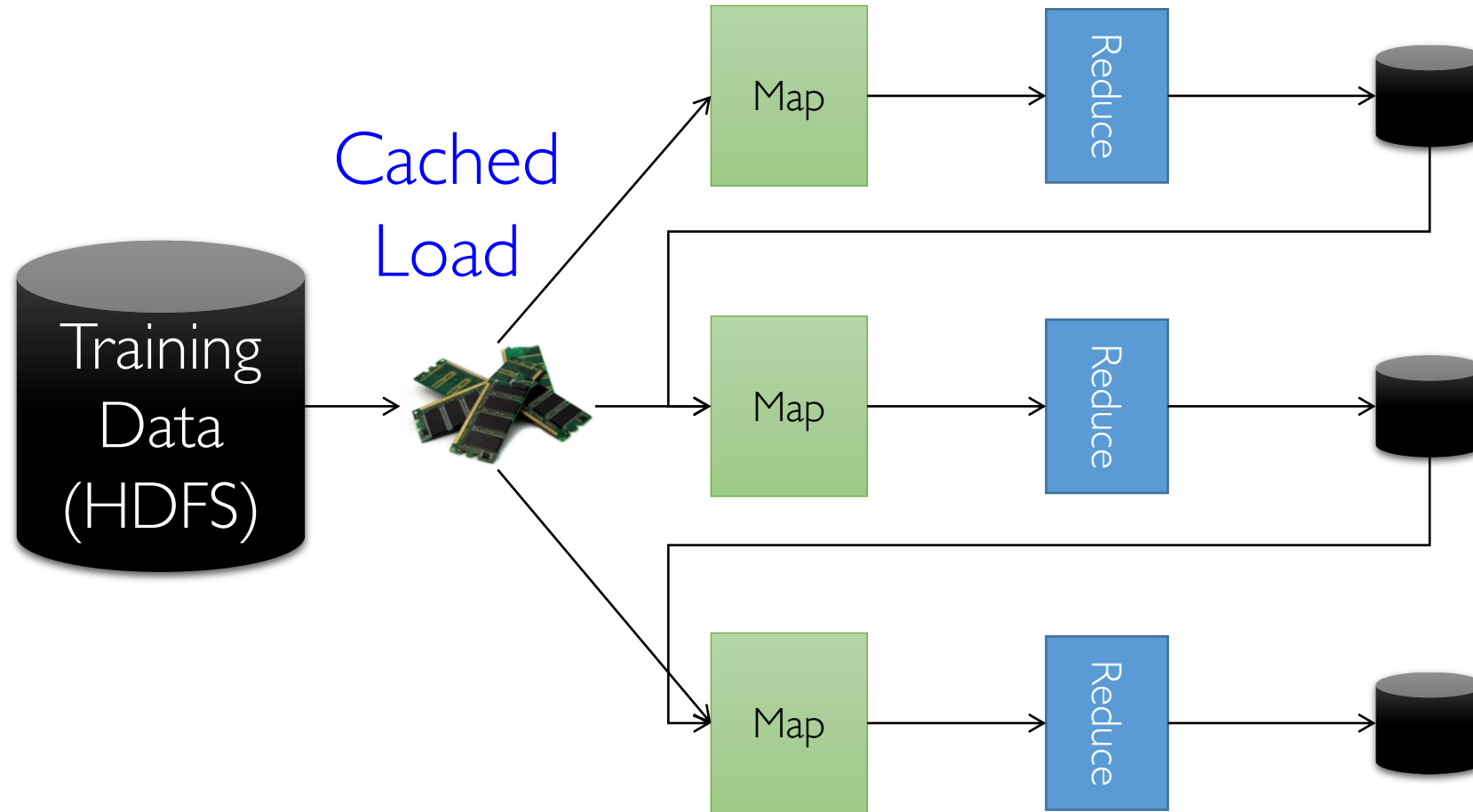
Implementation Details: Statistical Query Pattern

- **Iterative** ML → Data caching is important
 - Motivation behind Spark project

Map Reduce Dataflow View

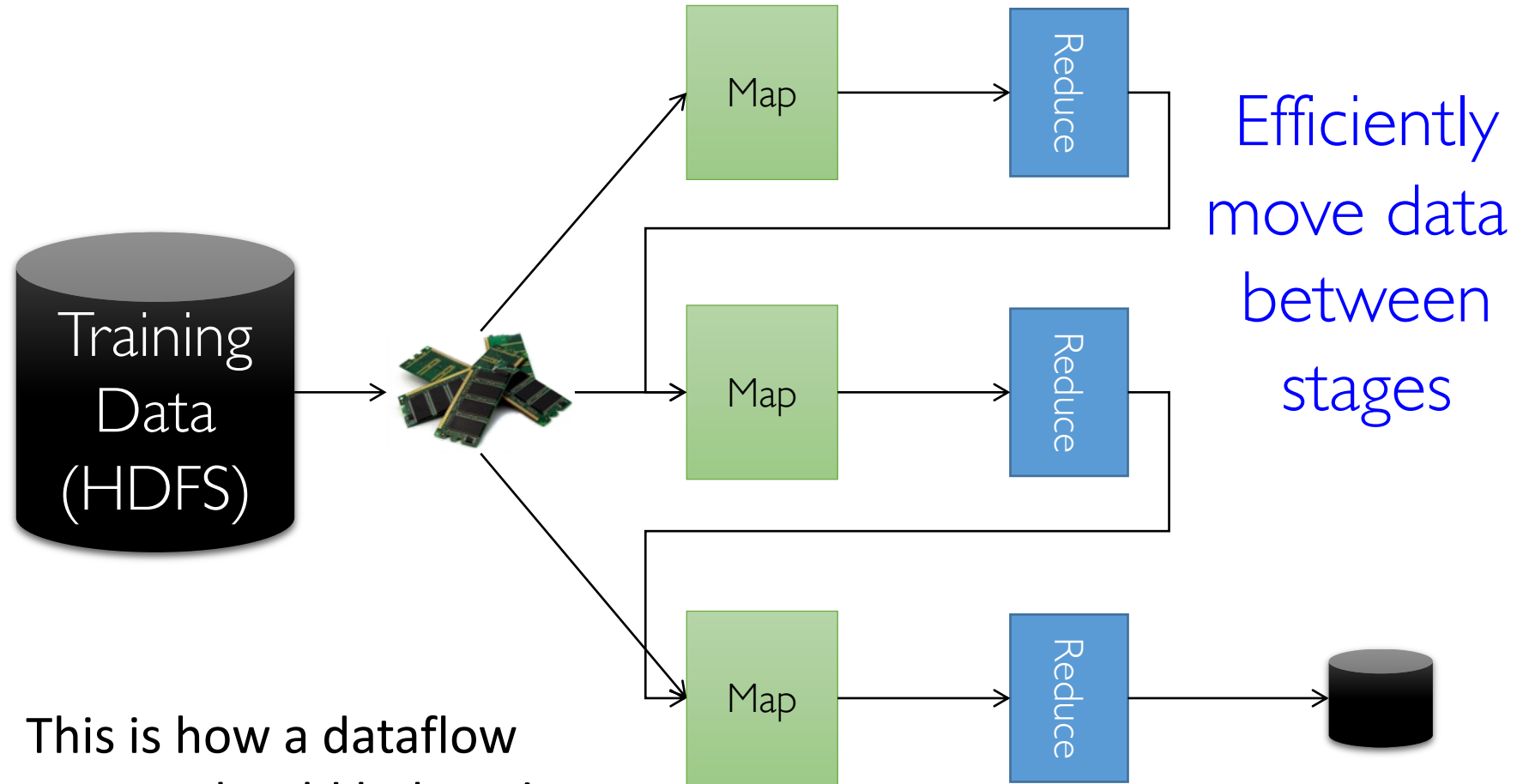


Spark Opt. Dataflow



10-100× faster than network and disk

Spark Opt. Dataflow View

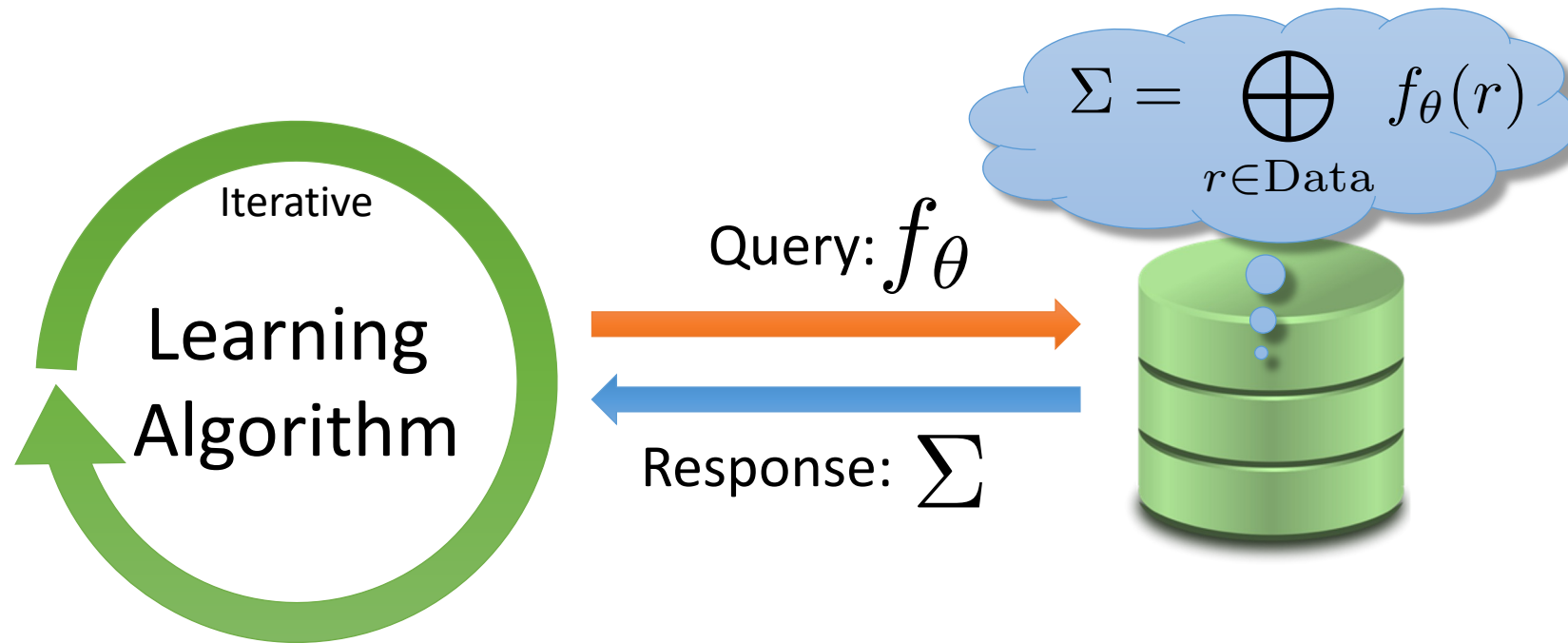


This is how a dataflow system should behave!

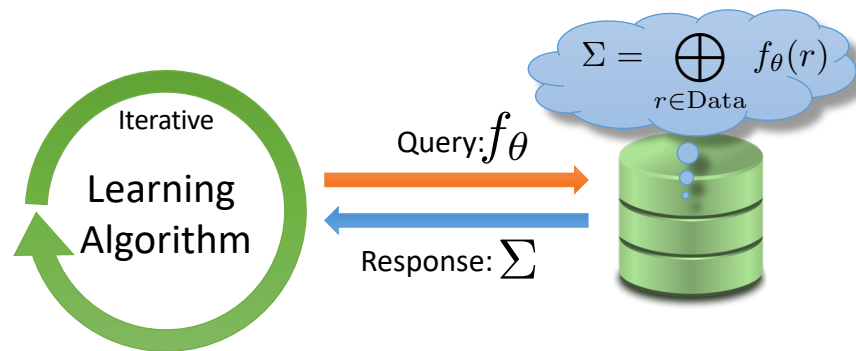
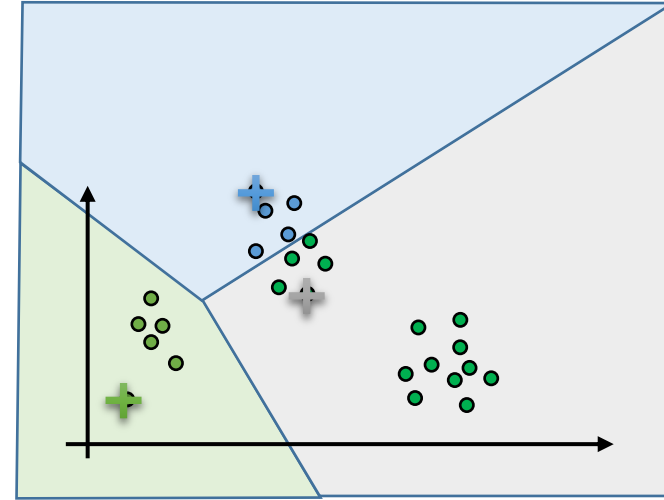
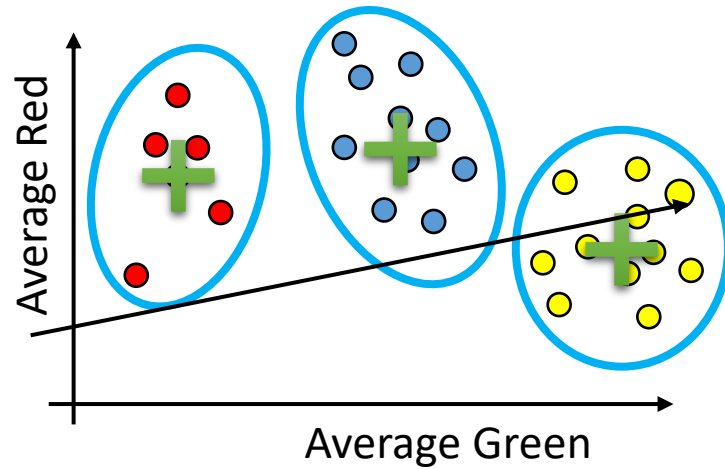
- What happened to map-reduce?

Implementation Details: Common Machine Learning Pattern

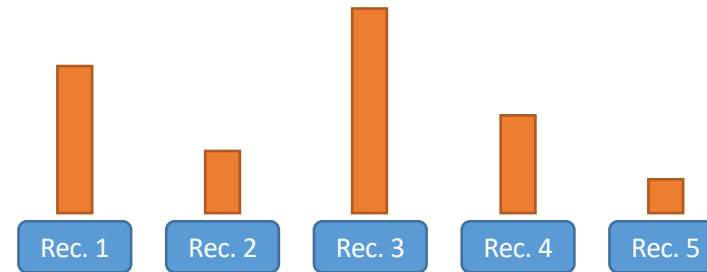
- Iterative ML → Data caching is important
 - Motivation behind Spark project
- Need to watch out for large θ and Σ



Summary of Clustering



```
SELECT nearest_UDF(centers, x) AS cid, mean_UDA(x)
FROM data GROUPBY cid
```





Taxonomy of Machine Learning

