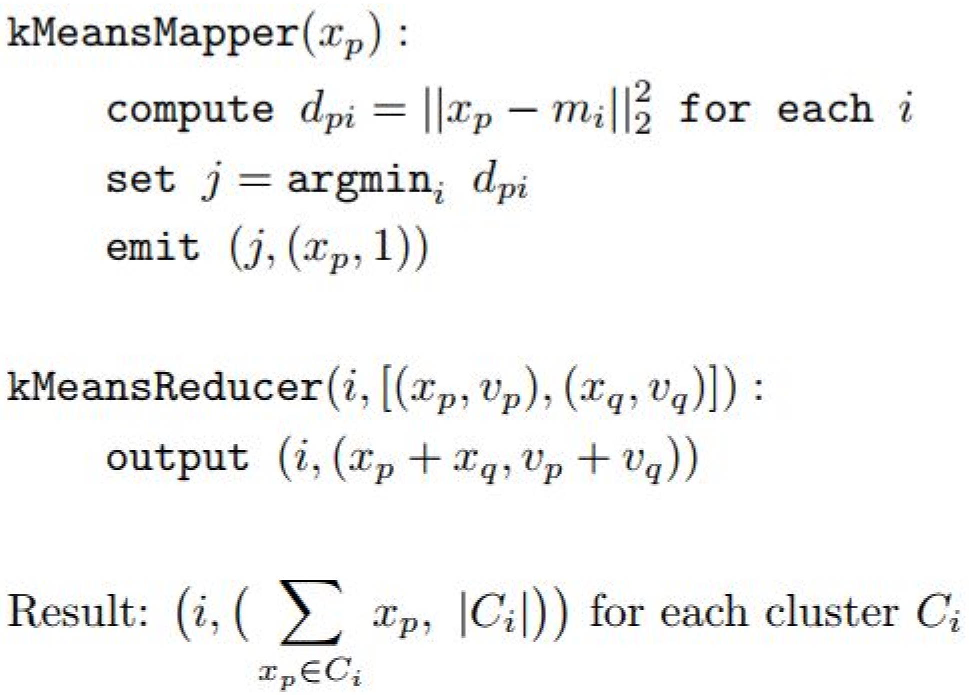
Map-Reduce Algorithms for *k*-means Clustering

Max Bodoia

k-means via MapReduce

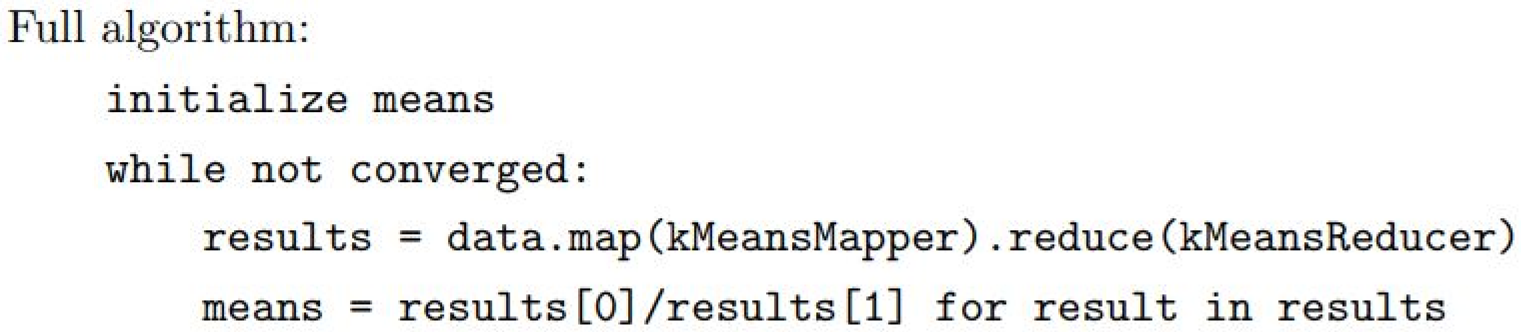


```
kMeansMapper($x_p$) :
    compute $d_{pi} = ||x_p - m_i||_2^2$ for each $i$
    set $j = \text{argmin}_i \; d_{pi}$
    emit $(j, (x_p, 1))$

kMeansReducer($i, [(x_p, v_p), (x_q, v_q)]$) :
    output $(i, (x_p + x_q, v_p + v_q))$
```

Result: $(i, (\sum_{x_p \in C_i} x_p, \; |C_i|))$ for each cluster C_i

k-means via MapReduce



Full algorithm:

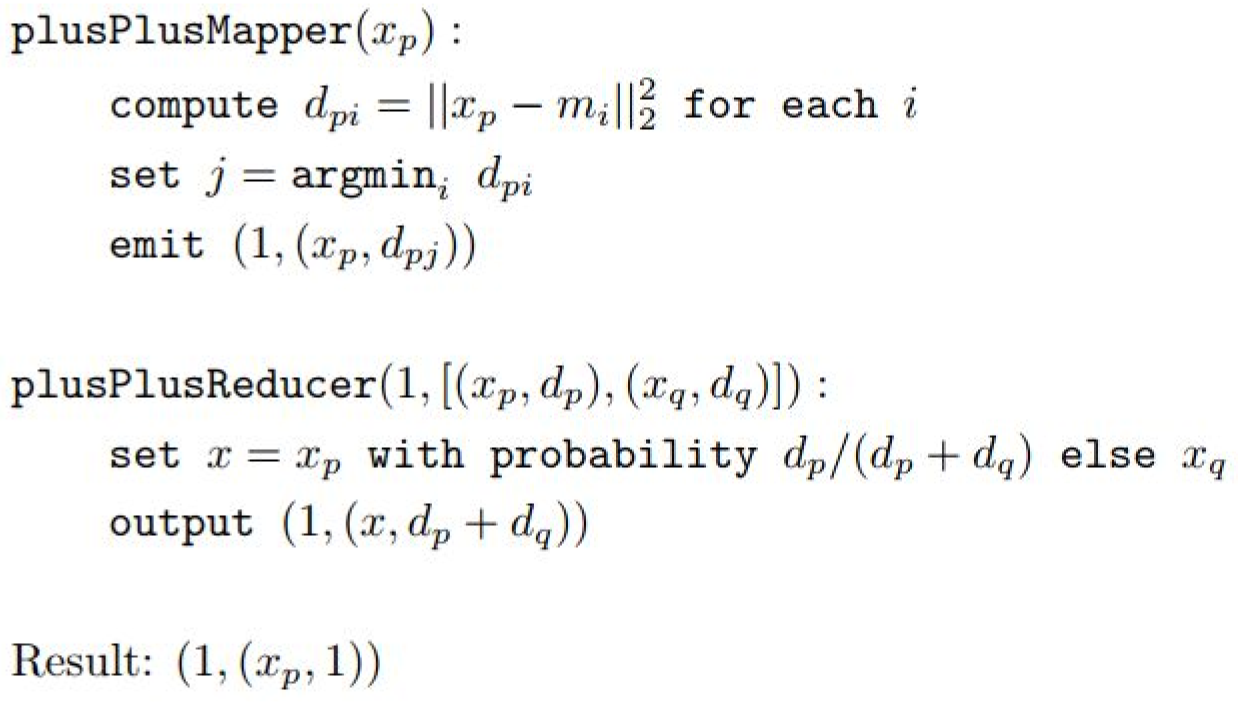
```
initialize means
while not converged:
    results = data.map(kMeansMapper).reduce(kMeansReducer)
    means = results[0]/results[1] for result in results
```

k-means++ via MapReduce

k-means++ Initialization (copied from Wikipedia):

1. Choose one center uniformly at random from among the data points.
2. For each data point x, compute D(x), the distance between x and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to $D(x)^2$.
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Now that the initial centers have been chosen, proceed using standard *k*-means clustering.

k-means++ via MapReduce

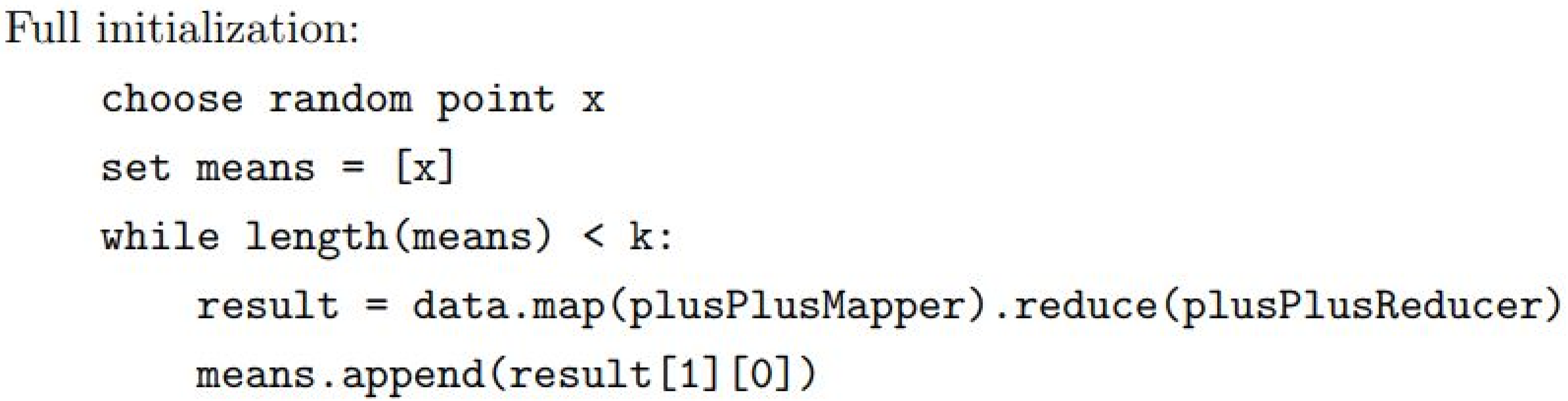


```
plusPlusMapper($x_p$):
    compute $d_{pi} = ||x_p - m_i||_2^2$ for each $i$
    set $j = \text{argmin}_i \; d_{pi}$
    emit $(1, (x_p, d_{pj}))$

plusPlusReducer($1, [(x_p, d_p), (x_q, d_q)]$):
    set $x = x_p$ with probability $d_p/(d_p + d_q)$ else $x_q$
    output $(1, (x, d_p + d_q))$

Result: $(1, (x_p, 1))$
```

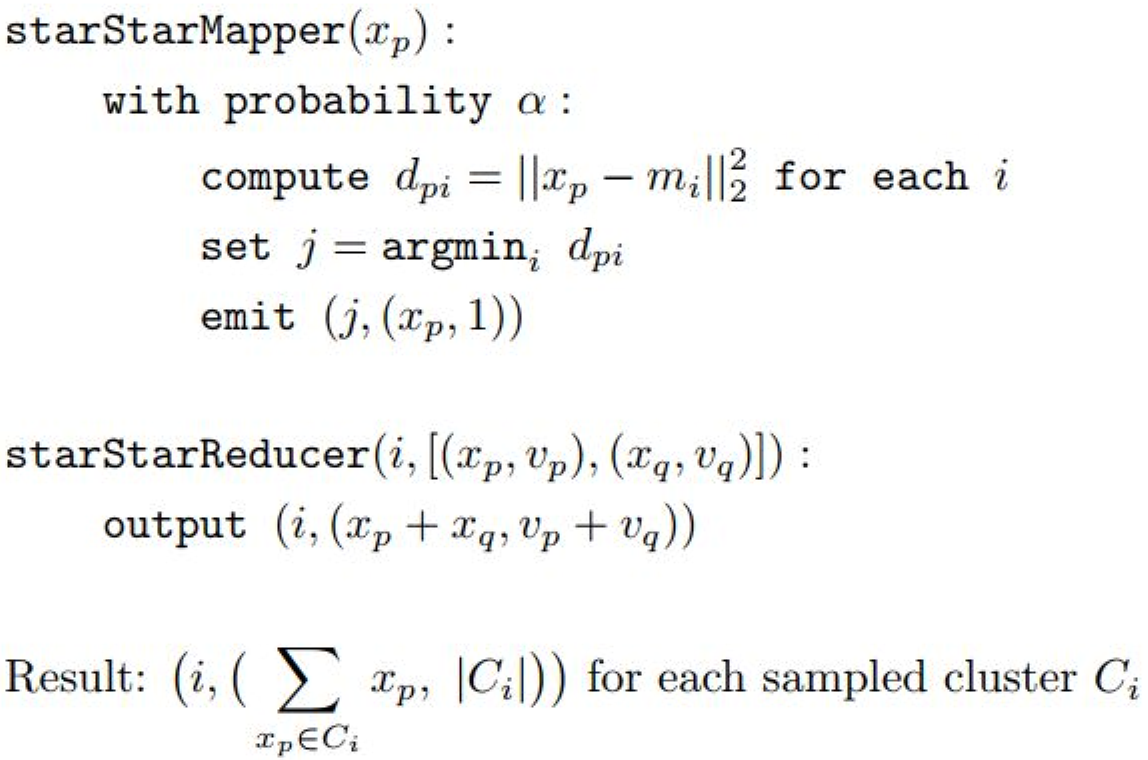
k-means++ via MapReduce



Full initialization:

```
    choose random point x
    set means = [x]
    while length(means) < k:
        result = data.map(plusPlusMapper).reduce(plusPlusReducer)
        means.append(result[1][0])
```

k-means** via MapReduce

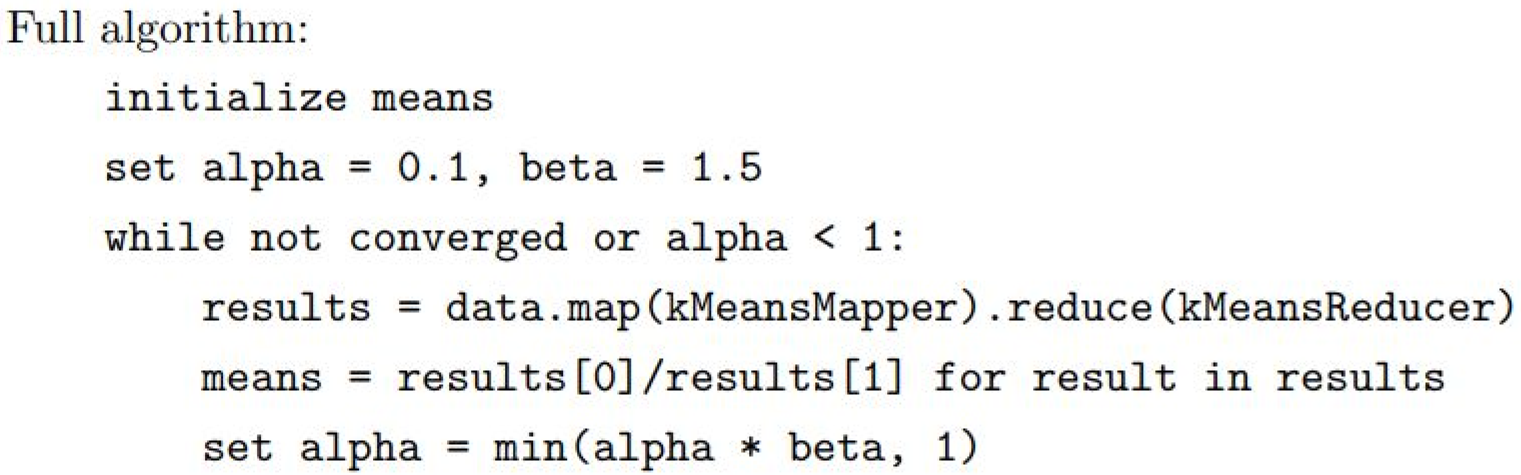


```
starStarMapper($x_p$) :
    with probability $\alpha$ :
        compute $d_{pi} = ||x_p - m_i||_2^2$ for each $i$
        set $j = \mathtt{argmin}_i \; d_{pi}$
        emit $(j, (x_p, 1))$

starStarReducer($i, [(x_p, v_p), (x_q, v_q)]$) :
    output $(i, (x_p + x_q, v_p + v_q))$
```

Result: $\left(i, \left(\sum_{x_p \in C_i} x_p, \; |C_i| \right)\right)$ for each sampled cluster C_i

k-means** via MapReduce



Full algorithm:

```
initialize means
set alpha = 0.1, beta = 1.5
while not converged or alpha < 1:
    results = data.map(kMeansMapper).reduce(kMeansReducer)
    means = results[0]/results[1] for result in results
    set alpha = min(alpha * beta, 1)
```

Empirical Results

	k-means	*k*-means++	*k*-means**
Average Error	181.9	**106.1**	252.7
Minimum Error	103.7	**99.8**	108.7
Average Time	3003	4375	**1705**

n = 2.5 million, d = 68, k = 10

Thank you!