Data clustering & the k-means algorithm

Lily Silverstein

April 27, 2016
Why clustering?

Unsupervised Learning

- Underlying structure
  - gain insight into data
  - generate hypotheses
  - detect anomalies
  - identify features
- Natural classification
  - e.g. biological organisms (phylogenetic relationships)
- Data compression
k-means

- popular for more than 50 years
- simple
- efficient
- empirically successful
k-means

Input:

- \( n \) points in \( \mathbb{R}^d \), \( X = \{x_1, \ldots, x_n\} \)
- \( k \), the number of clusters we want to partition \( X \) into
k-means

Input:
- $n$ points in $\mathbb{R}^d$, $X = \{x_1, \ldots, x_n\}$
- $k$, the number of clusters we want to partition $X$ into

Output:
- a partition that minimizes the squared error between the mean of each cluster and the points in that cluster.
More precisely, let $X = \{x_1, \ldots, x_n\}$ where $x_j \in \mathbb{R}^d$ for all $1 \leq j \leq n$.

Let $C = \{c_1, \ldots, c_k\}$ be $k$ clusters, each containing some of the $x_j$’s. Let $\mu_i$ be the mean of cluster $c_i$.

$$\mu_i = \frac{1}{\# c_i} \sum_{x_j \in c_i} x_j.$$
k-means

We define the squared error between $\mu_i$ and the points in $c_i$ by

$$E(c_i) = \sum_{x_j \in c_i} \|x_j - \mu_i\|^2.$$
**k-means**

We define the squared error between $\mu_i$ and the points in $c_i$ by

$$E(c_i) = \sum_{x_j \in c_i} \|x_j - \mu_i\|^2.$$

The sum of the squared error over all $k$ clusters is

$$E(C) = \sum_{i=1}^{k} \sum_{x_j \in c_i} \|x_j - \mu_i\|^2.$$
k-means

We define the squared error between $\mu_i$ and the points in $c_i$ by

$$E(c_i) = \sum_{x_j \in c_i} \| x_j - \mu_i \|^2.$$ 

The sum of the squared error over all $k$ clusters is

$$E(C) = \sum_{i=1}^{k} \sum_{x_j \in c_i} \| x_j - \mu_i \|^2.$$ 

This is the objective function that the algorithm is designed to minimize.
k-means in action

Example of the algorithm for points in $\mathbb{R}^2$, with $k = 3$. 
k-means in action (images taken from [Jain, 2010])

(a) Input data
k-means in action (images taken from [Jain, 2010])

(b) Seed point selection
k-means in action (images taken from [Jain, 2010])
k-means in action (images taken from [Jain, 2010])

(d) Iteration 3
k-means in action (images taken from [Jain, 2010])

(e) Final clustering
basic procedure

1. Select an initial set of $k$ means (for example, choose $k$ points from the dataset).
2. Assign each point to its closest mean to generate a new partition of the data.
3. Calculate the new set of $k$ means with respect to this partition.
4. Repeat Steps 2 and 3 until cluster membership stabilizes.
Will this procedure terminate?
k-means

Will this procedure terminate?
- Monotonely decreasing sequence of sum of squared errors
- Finite number of clusterings for finite point set $X$
Will this procedure terminate?

- Monotonely decreasing sequence of sum of squared errors
- Finite number of clusterings for finite point set $X$

Number of steps bounded by $O(n^{O(dk)})$  \textit{Inaba et al. 1994}
k-means

Will this procedure terminate?

- Monotonely decreasing sequence of sum of squared errors
- Finite number of clusterings for finite point set $X$

Number of steps bounded by $O(n^{O(dk)})$ \textit{Inaba et al. 1994}

$k$-means is a \textbf{greedy algorithm}. It may terminate in a local minimum.
What’s wrong with this example?
determining $k$

Sometimes we don’t know what $k$ should be \textit{a priori}.
determining $k$

Sometimes we don’t know what $k$ should be *a priori*.

Increasing $k$ will always decrease squared error! In fact for $k = n$ (the number of data points) the sum of squared errors is 0. So squared error does not tell us which $k$ to use.
Sometimes we don’t know what \( k \) should be \textit{a priori}.

Increasing \( k \) will always decrease squared error! In fact for \( k = n \) (the number of data points) the sum of squared errors is 0. So squared error does not tell us which \( k \) to use.

Idea: determine \( k \) based on the information we want from the clustering or run \( k \)-means for various \( k \), then use some heuristic to compare the results.
determining $k$

One heuristic: the *elbow method*.

[Image from http://commons.wikimedia.org/wiki/File:DataClustering ElbowCriterion.JPG]
in Matlab

```matlab
[idx,C] = kmeans(X,k);
```

$X$ is an array whose rows are the points in your dataset.

$$X = \begin{bmatrix}
-x_1 \\
-x_2 \\
\vdots \\
-x_n \\
\end{bmatrix}$$
in Matlab

\[ [\text{idx}, C] = \text{kmeans}(X, k); \]

\( k \) is the number of clusters.
in Matlab

\[ \text{[idx, C]} = \text{kmeans}(X, k); \]

*idx* is a vector of length *n* identifying which cluster each point belongs to.

\[ \text{idx} = \begin{bmatrix}
\text{cluster that } x_1 \text{ is in} \\
\text{cluster that } x_2 \text{ is in} \\
\vdots \\
\text{cluster that } x_n \text{ is in}
\end{bmatrix} \]
Clustering in Matlab

\[ [\text{idx}, C] = \text{kmeans}(X,k); \]

\( C \) is an array containing the \( k \) centroids (means).

\[
C = \begin{bmatrix}
\text{centroid of cluster 1} \\
\text{centroid of cluster 2} \\
\vdots \\
\text{centroid of cluster } k
\end{bmatrix}
\]
References