Clustering and Linear Independence

Pulkit Tandon

ENGR108
Stanford University

slides adopted from Prof. Stephen Boyd

July 5, 2022
Outline

Clustering

Algorithm

Examples

Applications
Clustering

- given $N$ $n$-vectors $x_1, \ldots, x_N$
- goal: partition (divide, cluster) into $k$ groups
- want vectors in the same group to be close to one another
Example settings

- topic discovery and document classification
  - $x_i$ is word count histogram for document $i$

- patient clustering
  - $x_i$ are patient attributes, test results, symptoms

- customer market segmentation
  - $x_i$ is purchase history and other attributes of customer $i$

- color compression of images
  - $x_i$ are RGB pixel values

- financial sectors
  - $x_i$ are $n$-vectors of financial attributes of company $i$
Clustering objective

- $G_j \subset \{1, \ldots, N\}$ is group $j$, for $j = 1, \ldots, k$
- $c_i$ is group that $x_i$ is in: $i \in G_{c_i}$
- group *representatives*: $n$-vectors $z_1, \ldots, z_k$

- clustering objective is

\[
= \frac{1}{N} \sum_{i=1}^{N} \|x_i - z_{c_i}\|^2
\]

mean square distance from vectors to associated representative

- small means good clustering

- goal: choose clustering $c_i$ and representatives $z_j$ to minimize
Partitioning the vectors given the representatives

- suppose representatives $z_1, \ldots, z_k$ are given
- how do we assign the vectors to groups, i.e., choose $c_1, \ldots, c_N$?

- $c_i$ only appears in term $\|x_i - z_{c_i}\|^2$ in
- to minimize over $c_i$, choose $c_i$ so $\|x_i - z_{c_i}\|^2 = \min_j \|x_i - z_j\|^2$
- *i.e., assign each vector to its nearest representative*
Choosing representatives given the partition

- given the partition $G_1, \ldots, G_k$, how do we choose representatives $z_1, \ldots, z_k$ to minimize?

- splits into a sum of $k$ sums, one for each $z_j$:

\[
= J_1 + \cdots + J_k, \quad J_j = \frac{1}{N} \sum_{i \in G_j} \|x_i - z_j\|^2
\]

- so we choose $z_j$ to minimize mean square distance to the points in its partition

- this is the mean (or average or centroid) of the points in the partition:

\[
z_j = \frac{1}{|G_j|} \sum_{i \in G_j} x_i
\]
**k-means algorithm**

- alternate between updating the partition, then the representatives
- a famous algorithm called *k-means*
- objective decreases in each step

---

given $x_1, \ldots, x_N \in \mathbb{R}^n$ and $z_1, \ldots, z_k \in \mathbb{R}^n$

**repeat**

*Update partition:* assign $i$ to $G_j$, $j = \arg \min_j \|x_i - z_j'\|^2$

*Update centroids:* $z_j = \frac{1}{|G_j|} \sum_{i \in G_j} x_i$

**until** $z_1, \ldots, z_k$ stop changing

---
Convergence of $k$-means algorithm

- goes down in each step, until the $z_j$’s stop changing
- but (in general) the $k$-means algorithm does not find the partition that minimizes

- $k$-means is a heuristic: it is not guaranteed to find the smallest possible value of
- the final partition (and its value of $J$) can depend on the initial representatives
- common approach:
  - run $k$-means 10 times, with different (often random) initial representatives
  - take as final partition the one with the smallest value of
Outline

Clustering

Algorithm

Examples

Applications
Iteration 1
Iteration 2

Examples
Outline

Clustering

Algorithm

Examples

Applications
Handwritten digit image set

- MNIST images of handwritten digits (via Yann Lecun)
- \( N = 60,000 \) 28 \( \times \) 28 images, represented as 784-vectors \( x_i \)
- 25 examples shown below

\[
\begin{array}{cccccc}
5 & 0 & 4 & 1 & 9 \\
2 & 1 & 3 & 1 & 4 \\
3 & 5 & 3 & 6 & 1 \\
7 & 2 & 8 & 6 & 9 \\
4 & 0 & 9 & 1 & 1 \\
\end{array}
\]
\textit{k-means image clustering}

\begin{itemize}
  \item $k = 20$, run 20 times with different initial assignments
  \item convergence shown below (including best and worst)
\end{itemize}
Group representatives, best clustering

<table>
<thead>
<tr>
<th>8</th>
<th>3</th>
<th>0</th>
<th>7</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>1</td>
<td>9</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>9</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6</td>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>
Topic discovery

- $N = 500$ Wikipedia articles, word count histograms with $n = 4423$
- $k = 9$, run 20 times with different initial assignments
- convergence shown below (including best and worst)
Topics discovered (clusters 1–3)

- words with largest representative coefficients

<table>
<thead>
<tr>
<th>Cluster 1</th>
<th>Cluster 2</th>
<th>Cluster 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>fight</td>
<td>0.038</td>
<td>holiday</td>
</tr>
<tr>
<td>win</td>
<td>0.022</td>
<td>celebrate</td>
</tr>
<tr>
<td>event</td>
<td>0.019</td>
<td>festival</td>
</tr>
<tr>
<td>champion</td>
<td>0.015</td>
<td>celebration</td>
</tr>
<tr>
<td>fighter</td>
<td>0.015</td>
<td>calendar</td>
</tr>
</tbody>
</table>

- titles of articles closest to cluster representative

Applications 24
Topics discovered (clusters 4–6)

- words with largest representative coefficients

<table>
<thead>
<tr>
<th>Cluster 4</th>
<th>Cluster 5</th>
<th>Cluster 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>album</td>
<td>0.031</td>
<td>game</td>
</tr>
<tr>
<td>release</td>
<td>0.016</td>
<td>season</td>
</tr>
<tr>
<td>song</td>
<td>0.015</td>
<td>team</td>
</tr>
<tr>
<td>music</td>
<td>0.014</td>
<td>win</td>
</tr>
<tr>
<td>single</td>
<td>0.011</td>
<td>player</td>
</tr>
</tbody>
</table>

- titles of articles closest to cluster representative
Topics discovered (clusters 7–9)

- Words with largest representative coefficients

<table>
<thead>
<tr>
<th>Cluster 7</th>
<th>Cluster 8</th>
<th>Cluster 9</th>
</tr>
</thead>
<tbody>
<tr>
<td>match</td>
<td>0.065</td>
<td>film</td>
</tr>
<tr>
<td>win</td>
<td>0.018</td>
<td>star</td>
</tr>
<tr>
<td>championship</td>
<td>0.016</td>
<td>role</td>
</tr>
<tr>
<td>team</td>
<td>0.015</td>
<td>play</td>
</tr>
<tr>
<td>event</td>
<td>0.015</td>
<td>series</td>
</tr>
</tbody>
</table>

- Titles of articles closest to cluster representative


Applications
Linear independence
Outline

Linear independence

Basis

Orthonormal vectors

Gram–Schmidt algorithm
set of \( n \)-vectors \( \{a_1, \ldots, a_k\} \) (with \( k \geq 1 \)) is *linearly dependent* if

\[
\beta_1 a_1 + \cdots + \beta_k a_k = 0
\]

holds for some \( \beta_1, \ldots, \beta_k \), that are not all zero

- equivalent to: at least one \( a_i \) is a linear combination of the others
- we say ‘\( a_1, \ldots, a_k \) are linearly dependent’

\( \{a_1\} \) is linearly dependent only if \( a_1 = 0 \)

\( \{a_1, a_2\} \) is linearly dependent only if one \( a_i \) is a multiple of the other

for more than two vectors, there is no simple to state condition
Example

- the vectors

\[
\begin{align*}
a_1 &= \begin{bmatrix} 0.2 \\ -7 \\ 8.6 \end{bmatrix}, &
\quad a_2 &= \begin{bmatrix} -0.1 \\ 2 \\ -1 \end{bmatrix}, &
\quad a_3 &= \begin{bmatrix} 0 \\ -1 \\ 2.2 \end{bmatrix}
\end{align*}
\]

are linearly dependent, since \( a_1 + 2a_2 - 3a_3 = 0 \)

- can express any of them as linear combination of the other two, e.g.,

\[
a_2 = (-1/2)a_1 + (3/2)a_3
\]
Linear independence

- set of $n$-vectors \( \{a_1, \ldots, a_k\} \) (with $k \geq 1$) is *linearly independent* if it is not linearly dependent, *i.e.*, 
\[
\beta_1 a_1 + \cdots + \beta_k a_k = 0
\]
holds only when $\beta_1 = \cdots = \beta_k = 0$

- we say ‘$a_1, \ldots, a_k$ are linearly independent’

- equivalent to: no $a_i$ is a linear combination of the others

- example: the unit $n$-vectors $e_1, \ldots, e_n$ are linearly independent
Linear combinations of linearly independent vectors

▶ suppose \( x \) is linear combination of linearly independent vectors \( a_1, \ldots, a_k \):

\[
x = \beta_1 a_1 + \cdots + \beta_k a_k
\]

▶ the coefficients \( \beta_1, \ldots, \beta_k \) are unique, i.e., if

\[
x = \gamma_1 a_1 + \cdots + \gamma_k a_k
\]

then \( \beta_i = \gamma_i \) for \( i = 1, \ldots, k \)

▶ this means that (in principle) we can deduce the coefficients from \( x \)

▶ to see why, note that

\[
(\beta_1 - \gamma_1) a_1 + \cdots + (\beta_k - \gamma_k) a_k = 0
\]

and so (by linear independence) \( \beta_1 - \gamma_1 = \cdots = \beta_k - \gamma_k = 0 \)
Outline

Linear independence

Basis

Orthonormal vectors

Gram–Schmidt algorithm
Independence-dimension inequality

- a linearly independent set of $n$-vectors can have at most $n$ elements
- put another way: any set of $n + 1$ or more $n$-vectors is linearly dependent
a set of \( n \) linearly independent \( n \)-vectors \( a_1, \ldots, a_n \) is called a \textit{basis}

any \( n \)-vector \( b \) can be expressed as a linear combination of them:

\[
b = \beta_1 a_1 + \cdots + \beta_n a_n
\]

for some \( \beta_1, \ldots, \beta_n \)

and these coefficients are unique

formula above is called \textit{expansion of} \( b \) \textit{in the} \( a_1, \ldots, a_n \) \textit{basis}

example: \( e_1, \ldots, e_n \) is a basis, expansion of \( b \) is

\[
b = b_1 e_1 + \cdots + b_n e_n
\]
Outline

Linear independence

Basis

Orthonormal vectors

Gram–Schmidt algorithm
Orthonormal vectors

- set of $n$-vectors $a_1, \ldots, a_k$ are (mutually) orthogonal if $a_i \perp a_j$ for $i \neq j$
- they are normalized if $\|a_i\| = 1$ for $i = 1, \ldots, k$
- they are orthonormal if both hold
- can be expressed using inner products as
  \[
  a_i^T a_j = \begin{cases} 
  1 & i = j \\
  0 & i \neq j 
  \end{cases}
  \]
- orthonormal sets of vectors are linearly independent
- by independence-dimension inequality, must have $k \leq n$
- when $k = n$, $a_1, \ldots, a_n$ are an orthonormal basis
Examples of orthonormal bases

- standard unit $n$-vectors $e_1, \ldots, e_n$

- the 3-vectors

$$
\begin{bmatrix}
0 \\
0 \\
-1
\end{bmatrix}, \quad \frac{1}{\sqrt{2}} \begin{bmatrix}
1 \\
1 \\
0
\end{bmatrix}, \quad \frac{1}{\sqrt{2}} \begin{bmatrix}
1 \\
-1 \\
0
\end{bmatrix}
$$

- the 2-vectors shown below
Orthonormal expansion

if $a_1, \ldots, a_n$ is an orthonormal basis, we have for any $n$-vector $x$

$$x = (a_1^T x)a_1 + \cdots + (a_n^T x)a_n$$

called orthonormal expansion of $x$ (in the orthonormal basis)

to verify formula, take inner product of both sides with $a_i$
Outline

Linear independence

Basis

Orthonormal vectors

Gram–Schmidt algorithm
Gram–Schmidt (orthogonalization) algorithm

- an algorithm to check if $a_1, \ldots, a_k$ are linearly independent
- we’ll see later it has many other uses
Gram–Schmidt algorithm

given $n$-vectors $a_1, \ldots, a_k$

for $i = 1, \ldots, k$

1. Orthogonalization: $\tilde{q}_i = a_i - (q_1^T a_i)q_1 - \cdots - (q_{i-1}^T a_i)q_{i-1}$

2. Test for linear dependence: if $\tilde{q}_i = 0$, quit

3. Normalization: $q_i = \tilde{q}_i / \|\tilde{q}_i\|

if G–S does not stop early (in step 2), $a_1, \ldots, a_k$ are linearly independent

if G–S stops early in iteration $i = j$, then $a_j$ is a linear combination of $a_1, \ldots, a_{j-1}$ (so $a_1, \ldots, a_k$ are linearly dependent)
Example

Gram–Schmidt algorithm
Analysis

let’s show by induction that $q_1, \ldots, q_i$ are orthonormal

▶ assume it’s true for $i - 1$

▶ orthogonalization step ensures that

$$\tilde{q}_i \perp q_1, \ldots, \tilde{q}_i \perp q_{i-1}$$

▶ to see this, take inner product of both sides with $q_j, j < i$

$$q_j^T \tilde{q}_i = q_j^T a_i - (q_1^T a_i)(q_j^T q_1) - \cdots - (q_{i-1}^T a_i)(q_j^T q_{i-1})$$

$$= q_j^T a_i - q_j^T a_i = 0$$

▶ so $q_i \perp q_1, \ldots, q_i \perp q_{i-1}$

▶ normalization step ensures that $\|q_i\| = 1$

Gram–Schmidt algorithm
assuming G–S has not terminated before iteration $i$

$\triangleright$ $a_i$ is a linear combination of $q_1, \ldots, q_i$:

$$a_i = \|q_i\|q_i + (q_1^T a_i)q_1 + \cdots + (q_{i-1}^T a_i)q_{i-1}$$

$\triangleright$ $q_i$ is a linear combination of $a_1, \ldots, a_i$: by induction on $i$,

$$q_i = \left(1/\|q_i\|\right) \left( a_i - (q_1^T a_i)q_1 - \cdots - (q_{i-1}^T a_i)q_{i-1} \right)$$

and (by induction assumption) each $q_1, \ldots, q_{i-1}$ is a linear combination of $a_1, \ldots, a_{i-1}$
suppose G–S terminates in step $j$

- $a_j$ is linear combination of $q_1, \ldots, q_{j-1}$
  \[ a_j = (q_1^T a_j)q_1 + \cdots + (q_{j-1}^T a_j)q_{j-1} \]

- and each of $q_1, \ldots, q_{j-1}$ is linear combination of $a_1, \ldots, a_{j-1}$

- so $a_j$ is a linear combination of $a_1, \ldots, a_{j-1}$
Complexity of Gram–Schmidt algorithm

- step 1 of iteration $i$ requires $i - 1$ inner products,
  
  $$q_1^T a_i, \ldots, q_{i-1}^T a_i$$

  which costs $(i - 1)(2n - 1)$ flops

- $2n(i - 1)$ flops to compute $\tilde{q}_i$

- $3n$ flops to compute $\|\tilde{q}_i\|$ and $q_i$

- total is

  $$\sum_{i=1}^{k} \left( (4n - 1)(i - 1) + 3n \right) = (4n - 1) \frac{k(k - 1)}{2} + 3nk \approx 2nk^2$$

  using $\sum_{i=1}^{k} (i - 1) = k(k - 1)/2$