Algorithm

1 Sketch Version

Input: $x_1 \ldots x_n \in \mathbb{R}^d$ ordered in decreasing frequency, number of clusters m
Output: hierarchical clustering of $x_1 \ldots x_n$

1. $A \leftarrow \{1, \ldots m\}$
2. For $i = 1 \ldots m$
   (a) $\text{twin}(i) \leftarrow \arg\min_{j \in A : j \neq i} \text{cost}(x_i, x_j)$
   (b) $\text{lb}(i) \leftarrow \min_{j \in A : j \neq i} \text{cost}(x_i, x_j)$
   (c) $\text{tight}(i) \leftarrow 1$
3. For $i = m + 1 \ldots 2n - 1$
   (a) If $i \leq n$
      i. $A \leftarrow A \cup \{i\}$
      ii. $\text{twin}(i) \leftarrow \arg\min_{j \in A : j \neq i} \text{cost}(x_i, x_j)$
      iii. $\text{lb}(i) \leftarrow \min_{j \in A : j \neq i} \text{cost}(x_i, x_j)$
      iv. $\text{tight}(i) \leftarrow 1$
      v. For $j \in A$, if $\text{cost}(x_i, x_j) < \text{lb}(j)$, set $\text{twin}(j) \leftarrow i$, $\text{lb}(i) \leftarrow \text{cost}(x_i, x_j)$, $\text{tight}(j) \leftarrow 1$
   (b) $a \leftarrow \arg\min_{j \in A} \text{lb}(j)$
   (c) While $\text{tight}(a) \neq 1$
      i. $\text{twin}(a) \leftarrow \arg\min_{j \in A : j \neq a} \text{cost}(x_a, x_j)$
      ii. $\text{lb}(a) \leftarrow \min_{j \in A : j \neq a} \text{cost}(x_a, x_j)$
      iii. $\text{tight}(a) \leftarrow 1$
      iv. $a \leftarrow \arg\min_{j \in A} \text{lb}(j)$
   (d) $b \leftarrow \text{twin}(a)$, $c \leftarrow i - m + n$
   (e) $x_c \leftarrow \text{merge}(x_a, x_b)$
   (f) $A \leftarrow A \cup \{c\} \backslash \{a, b\}$
   (g) $\text{twin}(c) \leftarrow \arg\min_{j \in A : j \neq c} \text{cost}(x_c, x_j)$
   (h) $\text{lb}(c) \leftarrow \min_{j \in A : j \neq c} \text{cost}(x_c, x_j)$
   (i) $\text{tight}(c) \leftarrow 1$
   (j) For $j \in A$, if $j \neq c$ and $\text{twin}(j) \in \{a, b\}$, set $\text{tight}(c) \leftarrow 0$

2 Implementation Version

We will have total $2n-1$ clusters, referred to by their "**ID**"s $1\dots 2n-1$:

- **ID** $\leq n$: clusters corresponding to words $1\dots n$
- **ID** $> n$: clusters corresponding to the $\mathbf{ID}^{th}$ merge

But we only need to keep around at most $m+1$ clusters at any point. To achieve this, we will use the following vectors of length $m+1$ which we call *holders*. The elements holders contain will change dynamically.

- $\mathtt{I}$: $\mathtt{I}[i] \in \{1\dots 2n-1\}$ is the cluster **ID** at position $i \in [m+1]$
- $\mathtt{C}$: $\mathtt{C}[i] \in \mathbb{R}^d$ is the center of cluster $\mathtt{I}[i]$
- lb: $\mathrm{lb}[i] \in \mathbb{R}$ is the lower bound of cluster $\mathtt{I}[i]$
- twin: $\mathrm{twin}[i] \in \{1\dots m+1\}$ and $\mathtt{I}[\mathrm{twin}[i]]$ is the twin cluster **ID** of cluster $\mathtt{I}[i]$
- tight: $\mathrm{tight}[i] \in \{0,1\}$ is the tightness of cluster $\mathtt{I}[i]$

We also keep a vector $\mathtt{S}$ of length $2n-1$ for recording the sizes of the clusters. In constrast to holders, it has static elements—once stored, a value in this vector does not change.

- $\mathtt{S}$: $\mathtt{S}[\mathbf{ID}] \in \{1\dots n\}$ is the size of cluster $\mathbf{ID} = 1\dots 2n-1$

Finally, the output of the algorithm will be recorded in matlab style as $\mathtt{Z} \in \mathbb{R}^{(n-1)\times 3}$ where $\mathtt{Z}(\mathbf{ID}-n) \in \mathbb{R}^3$ for $\mathbf{ID} = n+1\dots 2n-1$ corresponds to cluster **ID** and

- $\mathtt{Z}(\mathbf{ID}-n)_1$: left child of cluster **ID**
- $\mathtt{Z}(\mathbf{ID}-n)_2$: right child of cluster **ID**
- $\mathtt{Z}(\mathbf{ID}-n)_3$: merge cost of cluster **ID**

For $i, j \leq m+1$, we use function $\mathrm{cost}(i,j)$ to compute the distance between clusters $\mathtt{I}[i]$ and $\mathtt{I}[j]$. One call to cost has runtime $O(d)$.

$$\mathrm{cost}(i,j) = \frac{\mathtt{S}[\mathtt{I}[i]] \times \mathtt{S}[\mathtt{I}[j]]}{\mathtt{S}[\mathtt{I}[i]] + \mathtt{S}[\mathtt{I}[j]]} ||\mathtt{C}[i] - \mathtt{C}[j]||_2^2$$

Also, use function $\mathrm{merge}(i,j)$ to compute the new center of the result of merging clusters $\mathtt{I}[i]$ and $\mathtt{I}[j]$. One call to merge has runtime $O(d)$.

$$\mathrm{merge}(i,j) = \frac{\mathtt{S}[\mathtt{I}[i]]}{\mathtt{S}[\mathtt{I}[i]] + \mathtt{S}[\mathtt{I}[j]]}\mathtt{C}[i] + \frac{\mathtt{S}[\mathtt{I}[j]]}{\mathtt{S}[\mathtt{I}[i]] + \mathtt{S}[\mathtt{I}[j]]}\mathtt{C}[j]$$

2.1 Initialize the first m clusters: for $i = 1\dots m$

1. $\mathtt{S}[i] = 1$
2. $\mathtt{I}[i] = i$
3. $\mathtt{C}[i] = x_i$
4. $\mathrm{lb}[i] = \min_{j\in[m]:j\neq i} \mathrm{cost}(i,j)$
5. $\mathrm{twin}[i] = \arg\min_{j\in[m]:j\neq i} \mathrm{cost}(i,j)$
6. $\mathrm{tight}[i] = 1$

Note that this takes $O(dm^2)$ operations. Each holder has an empty slot at position $m+1$.

2.2 Perform $n-1$ merges: for $i = n+1 \dots 2n-1$

Let $t = i - n + m$. Thus

- $t \in [m+1, n]$ is the cluster **ID**s corresponding to the last $n-m$ words
- $i \in [n+1, 2n-1]$ is the cluster **ID**s corresponding to the $n-1$ merges

2.2.1 If $t \leq n$, do this extra step

1. $\texttt{S}[t] = 1$
2. $\texttt{I}[m+1] = t$
3. $\texttt{C}[m+1] = x_t$
4. $\text{lb}[m+1] = \min_{j \in [m]} \text{cost}(m+1, j)$
5. $\text{twin}[m+1] = \arg\min_{j \in [m]} \text{cost}(m+1, j)$
6. $\text{tight}[m+1] = 1$
7. For $j \in [m]$, if $\text{cost}(m+1, j) < \text{lb}[j]$, set $\text{lb}[j] \leftarrow \text{cost}(m+1, j)$, $\text{twin}[j] \leftarrow m+1$, $\text{tight}[j] \leftarrow 1$

This extra step takes $O(dm)$ operations. Now each holder has full $m+1$ elements.

2.2.2 Find the positions a and b of the closest pair

The clusters in consideration are the first $r := \min\{m+1, 2n-i\}$ elements in holders.[1]

- $a \leftarrow \arg\min_{j \in [r]} \text{lb}[j]$
- While $\text{tight}[a] \neq 1$
  1. $\text{lb}[a] = \min_{j \in [r]: j \neq a} \text{cost}(a, j)$
  2. $\text{twin}[a] = \arg\min_{j \in [r]: j \neq a} \text{cost}(a, j)$
  3. $\text{tight}[a] = 1$
  4. For $j \in [r]$, if $\text{cost}(a, j) < \text{lb}[j]$, set $\text{lb}[j] \leftarrow \text{cost}(a, j)$, $\text{twin}[j] \leftarrow a$, $\text{tight}[j] \leftarrow 1$
  5. $a \leftarrow \arg\min_{j \in [r]} \text{lb}[j]$
- $b \leftarrow \text{twin}[a]$
- Make sure $a < b$

This step takes $O(m)$ in the best case and $O(dm^2)$ in the worst case.

2.2.3 Record the merge between $\texttt{I}[a]$ and $\texttt{I}[b]$ as cluster i

1. $\texttt{Z}[i-n]_1 = \texttt{I}[a]$
2. $\texttt{Z}[i-n]_2 = \texttt{I}[b]$
3. $\texttt{Z}[i-n]_3 = \text{cost}(a, b)$

[1] At the $(n-m)^{th}$ merge (i.e., the first merge without adding a cluster), we will have $i = n + (n-m) = 2n - m$ so that $r = \min\{m+1, m\} = m$.

2.2.4 Put cluster i at position a in holders

1. $\texttt{S}[i] = \texttt{S}[\texttt{I}[a]] + \texttt{S}[\texttt{I}[b]]$
2. $\texttt{C}[a] = \text{merge}(a, b)$ // overwriting: now $\texttt{C}[a]$ is the center of cluster i
3. $\texttt{I}[a] = i$
4. $\text{lb}[a] = \min_{j\in[r]:j\neq a,b} \text{cost}(a, j)$
5. $\text{twin}[a] = \arg\min_{j\in[r]:j\neq a,b} \text{cost}(a, j)$
6. $\text{tight}[a] = 1$

2.2.5 Push the cluster at position b out of consideration: for $j = 1 \ldots r - 1$

- If $j < b$
  - If $\text{twin}[j] > b$, set $\text{twin}[j] = \text{twin}[j] - 1$
- If $j \geq b$
  1. $\texttt{I}[j] = I[j + 1]$
  2. $\texttt{C}[j] = \texttt{C}[j + 1]$
  3. $\text{lb}[j] = \text{lb}[j + 1]$
  4. $\text{twin}[j] = \text{twin}[j + 1]$ if $\text{twin}[j + 1] < b$, $\text{twin}[j] = \text{twin}[j + 1] - 1$ if $\text{twin}[j + 1] \geq b$!
  5. $\text{tight}[j] = \text{tight}[j + 1]$

Now the values at position r in our vectors are meaningless.

2.2.6 Loosen the bounds of twins of a and b: for $j = 1 \ldots r - 1$

- If $\text{twin}[j] \in \{a, b\}$, then set $\text{tight}[j] \leftarrow 0$

3 Implementation Version: Priority Queue

Keep a priority queue Q. We can build a queue containing N objects in $O(N)$. Then we can access an object with the smallest value in $O(1)$. (We can pop it and rearrange the queue in $O(\log N)$.) We can insert an object into the queue in $O(\log N)$. We will assume that the elements in Q are pairs (i, v) where i is the key and v is the value.

3.1 Initialize the first m clusters for $i = 1 \ldots m$

1. $\texttt{S}[i] = 1$
2. $\texttt{I}[i] = i$
3. $\texttt{C}[i] = x_i$
4. $\text{lb}[i] = \min_{j \in [m]: j \neq i} \text{cost}(i, j)$
5. $\text{twin}[i] = \arg\min_{j \in [m]: j \neq i} \text{cost}(i, j)$
6. $\text{tight}[i] = 1$

3.2 Put the m lower bounds in a queue Q

$$Q \leftarrow \text{build-queue}(\{(i, \text{lb}[i]) \text{ for } i = 1 \ldots m\})$$

Constructing it has runtime is $O(m)$.

3.3 Perform $n - 1$ merges: for $i = n + 1 \ldots 2n - 1$

Let $t = i - n + m$.

3.3.1 If $t \leq n$, do this extra step

1. $\texttt{S}[t] = 1$
2. $\texttt{I}[m + 1] = t$
3. $\texttt{C}[m + 1] = x_t$
4. $\text{lb}[m + 1] = \min_{j \in [m]} \text{cost}(m + 1, j)$
5. $\text{twin}[m + 1] = \arg\min_{j \in [m]} \text{cost}(m + 1, j)$
6. $\text{tight}[m + 1] = 1$
7. For $j \in [m]$, if $\text{cost}(m + 1, j) < \text{lb}[j]$, set $\text{lb}[j] \leftarrow \text{cost}(m + 1, j)$, $\text{twin}[j] \leftarrow m + 1$, $\text{tight}[j] \leftarrow 1$

This extra step takes $O(dm)$ operations. Now each holder has full $m+1$ elements. Since we've potentially changed many lower bounds, we have to reconstruct the queue:

$$Q \leftarrow \text{build-queue}(\{(i, \text{lb}[i]) \text{ for } i = 1 \ldots m + 1\})$$

Runtime is $O(m)$.

3.3.2 Find the positions a and b of the closest pair

We care about the first $r = \min\{m+1, 2n-i\}$ elements in holders.

- $(a, \text{lb}[a]) \leftarrow Q.\text{top}()$
- While $\text{tight}[a] \neq 1$
  1. $\text{lb}[a] = \min_{j \in [r]: j \neq a} \text{cost}(a, j)$
  2. $\text{twin}[a] = \arg\min_{j \in [r]: j \neq a} \text{cost}(a, j)$
  3. $\text{tight}[a] = 1$
  4. For $j \in [r]$, if $\text{cost}(a, j) < \text{lb}[j]$, set $\text{lb}[j] \leftarrow \text{cost}(a, j)$, $\text{twin}[j] \leftarrow a$, $\text{tight}[j] \leftarrow 1$
  5. Again, we've potentially changed many lower bounds, so we have to reconstruct the queue:
  $$Q \leftarrow \text{build-queue}(\{(i, \text{lb}[i]) \text{ for } i = 1 \ldots r\})$$
  6. $(a, \text{lb}[a]) \leftarrow Q.\text{top}()$
- $b \leftarrow \text{twin}[a]$
- Make sure $a < b$

This step now takes $O(1)$ in the best case. It's still $O(dm^2)$ in the worst case, but the difference is:

- Before: $O(m + m \times (dm + m)) = O(m^2 d)$
- Now: $O(1 + m \times (dm + m + 1)) = O(m^2 d)$

3.3.3 Record the merge between $\texttt{I}[a]$ and $\texttt{I}[b]$ as cluster i

1. $\texttt{Z}[i-n]_1 = \texttt{I}[a]$
2. $\texttt{Z}[i-n]_2 = \texttt{I}[b]$
3. $\texttt{Z}[i-n]_3 = \text{cost}(a, b)$

3.3.4 Put cluster i at position a in holders

1. $\texttt{S}[i] = \texttt{S}[\texttt{I}[a]] + \texttt{S}[\texttt{I}[b]]$
2. $\texttt{C}[a] = \text{merge}(a, b)$ // overwriting: now $\texttt{C}[a]$ is the center of cluster i
3. $\texttt{I}[a] = i$
4. $\text{lb}[a] = \min_{j \in [r]: j \neq a, b} \text{cost}(a, j)$
5. $\text{twin}[a] = \arg\min_{j \in [r]: j \neq a, b} \text{cost}(a, j)$
6. $\text{tight}[a] = 1$

3.3.5 Push the cluster at position b out of consideration: for $j = 1 \ldots r-1$

- If $j < b$
  - If $\text{twin}[j] > b$, set $\text{twin}[j] = \text{twin}[j] - 1$
- If $j \geq b$
  1. $\texttt{I}[j] = I[j+1]$
  2. $\texttt{C}[j] = \texttt{C}[j+1]$
  3. $\text{lb}[j] = \text{lb}[j+1]$
  4. $\text{twin}[j] = \text{twin}[j+1]$ if $\text{twin}[j+1] < b$, $\text{twin}[j] = \text{twin}[j+1] - 1$ if $\text{twin}[j+1] \geq b$!
  5. $\text{tight}[j] = \text{tight}[j+1]$

Now the values at position r in our vectors are meaningless.

3.3.6 Loosen the bounds of twins of a and b: for $j = 1 \ldots r-1$

- If $\mathrm{twin}[j] \in \{a, b\}$, then set $\mathrm{tight}[j] \leftarrow 0$

3.3.7 Rebuild the queue

$$Q \leftarrow \text{build-queue}(\{(i, \mathrm{lb}[i]) \text{ for } i = 1 \ldots r-1\})$$