Dynamic visualization for L1 fusion convex clustering in near-linear time (Supplementary material)

Bingyuan Zhang1
Jie Chen1
Yoshikazu Terada1,2

1 Graduate School of Engineering Science, Osaka University, Japan.
2 RIKEN Center for Advanced Intelligence Project (AIP)

1 AN EXAMPLE N = 3

Consider the case \( n = 3 \), given \( x_1 \leq x_2 \leq x_3 \), we want to minimize the following problem to obtain \( \hat{a}_1 \leq \hat{a}_2 \leq \hat{a}_3 \).

\[
(\hat{a}_1, \hat{a}_2, \hat{a}_3) = \arg \min_{(a_1, a_2, a_3)} \left\{ \frac{1}{2} [(x_1 - a_1)^2 + (x_2 - a_2)^2 + (x_3 - a_3)^2] + \lambda_1 |a_1 - a_2| + \lambda_2 |a_2 - a_3| \right\}
\]

Suppose \( \hat{a}_2 \) is known, by definition \( \hat{a}_1 \) is equal to:

\[
\hat{a}_1 = \arg \min_b \frac{1}{2} (x_1 - b)^2 + \lambda_1 |b - \hat{a}_2|
\]

\[
= \arg \min_b h_1(b) + \lambda_1 |b - \hat{a}_2|
\]

Since the \( b \) here represents \( \hat{a}_1 \) and is always smaller than \( \hat{a}_2 \). We only need to consider two cases:

- (1) \( b < \hat{a}_2 \). At that case, by KKT condition it is easy to find \( \hat{a}_1 = U_1 \).
- (2) \( b = \hat{a}_2 \). In other words, \( \hat{a}_1 = \hat{a}_2 \).

From that we get

\[
\hat{a}_1 = \arg \min_b h_1(b) + \lambda_1 |b - \hat{a}_2| = \max(\hat{a}_2, U_1)
\]

Similarly for \( \hat{a}_2 \), we suppose \( \hat{a}_3 \) is known:

\[
\hat{a}_2 = \arg \min_b \frac{1}{2} [(x_1 - \phi_1(b))^2 + (x_2 - b)^2] + \lambda_1 |\phi_1(b) - b| + \lambda_2 |b - \hat{a}_3|
\]

\[
= \arg \min_b h_2(b) + \lambda_2 |b - \hat{a}_3| = \max(\hat{a}_3, U_2)
\]

Next we need to find \( U_1, U_2 \) and \( \hat{a}_3 \), with whom \( \hat{a}_1 \) and \( \hat{a}_2 \) can be obtained immediately. We solve it in the following order:

\[
U_1 \rightarrow U_2 \rightarrow \hat{a}_3 \rightarrow \hat{a}_2 \rightarrow \hat{a}_1.
\]

For \( U_1 \), it is straightforward that \( U_1 = x_1 + \lambda_1 \). Then for \( U_2 \), the \( U_2 \) satisfies \( g_2(U_2) = \lambda_2 \), and \( g_2(b) \) is a continuous piecewise linear function shown in the figure:

\[
g_2(b) = \begin{cases} 
    g_1(b) & b \leq U_1 \n    \lambda_1 \mathbb{I}[b > U_1] + (b - x_2) & \text{otherwise}
\end{cases}
\]

Supplement for the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021).
Because $g_2$ is composed of two lines, the key is to locate which line is $(U_2, \lambda_2)$ on. According to the algorithm 2, we first search from the right by assuming the $(U_2, \lambda_2)$ is on the right line, and get $\beta = \lambda_2 - \lambda_1 + x_2$ which makes $(\beta, \lambda_2)$ the intersection point of $y = \lambda_2$ with the right line. Next we compare the $\beta$ with $U_1$ in figure 1. If $\beta \geq U_1$, $(U_2, \lambda_2)$ is indeed on the right part of the line, then we have $U_2 = \beta$; otherwise we update the slope and intercept to be those of the left line and let $U_2 = \beta_{\text{new}} = \lambda_2 + (x_1 + x_2)/2$.

Figure 1: An image of $g_2(b)$: the solid line is an example of $g_2(b)$ with a branch point $U_1$. The search starts to find the $\beta$ first and if $\beta$ does not qualify $\beta > U_1$, then we go to find the $\beta_{\text{new}}$.

This search process is efficient enough but is still not linear. To make the search from left to right more efficient, some care need to be taken.

Here we illustrate the erase step in the line 11 of the algorithm 2, we consider the case when $U_2 < U_1$. And now we want to find $\hat{a}_3$, and we also search from the right part of the $g_3$ function. The only difference between searching $U_2$ and $\hat{a}_3$ is we let $\beta$ satisfy $g_3(\beta) = 0$. We first compare the $\beta$ with $U_2$, and if $\beta < U_2$, the comparison between $\beta$ and $U_1$ is not necessary any more because we already know $U_2 < U_1$. Thus we can delete the $U_1$ after obtaining a $U_2$ that is smaller than $U_1$. In the end, all the $U_i$ can be deleted at most once: once a $U_j$, $j > i$ is found such that $U_j < U_i$, the former $U_i$ can be deleted immediately and never used again. By doing this the DP algorithm becomes much more efficiently and finally takes linear time.

2 SIMULATION DETAILS

Both standard errors and means over 30 replications are reported in the following table. When the sample size becomes larger, C-PAINT becomes faster than FUSION.

<table>
<thead>
<tr>
<th>Sample size $n$</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>5000</th>
<th>10000</th>
<th>50000</th>
</tr>
</thead>
<tbody>
<tr>
<td>CARPl1</td>
<td>0.32(2.7e-3)</td>
<td>232.3(69)</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>FLSA</td>
<td>0.04(5.2e-4)</td>
<td>3.4(4.8e-2)</td>
<td>35.6(0.4)</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>ADMM</td>
<td>0.07(9.3e-4)</td>
<td>4.9(4.8e-2)</td>
<td>50.8(20)</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>AMA</td>
<td>0.07(1.1e-3)</td>
<td>3.3(2.7e-2)</td>
<td>42.7(15)</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>FUSION</td>
<td>5e-4(9.2e-5)</td>
<td>4e-3(1.1e-4)</td>
<td>1.2e-2(2.0e-4)</td>
<td>0.33(3.1e-3)</td>
<td>1.1(7.4e-3)</td>
<td>27.7(2.8e-2)</td>
</tr>
<tr>
<td>C-PAINT</td>
<td>9.7e-4(2.6e-4)</td>
<td>3.8e-3(1.2e-4)</td>
<td>8.9e-3(1.5e-4)</td>
<td>0.15(7.5e-3)</td>
<td>0.41(1.1e-2)</td>
<td>5.6(4.2e-2)</td>
</tr>
</tbody>
</table>

Table 1: Run times Comparison. The means and standard errors of each method over 30 replications are reported. Here * means we cannot obtain the solutions within a reasonable time.