Implementation of Fast Biclustering Algorithm
Outline

• Theoretical Introduction
  • Clustering & Biclustering
  • Applications
  • Previous Attempts
  • Our algorithm

• High Performance Computing
  • Complexity
  • Problem Scale Matters
  • What makes our code fast.

• Results

• Further work
Clustering & Biclustering

- Clustering
- Suppose given the data below

## 1. REPUBLICAN VOTE FOR PRESIDENT

<table>
<thead>
<tr>
<th>State</th>
<th>00</th>
<th>04</th>
<th>08</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
<th>40</th>
<th>44</th>
<th>48</th>
<th>52</th>
<th>56</th>
<th>60</th>
<th>64</th>
<th>68</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama (AA)</td>
<td>35</td>
<td>21</td>
<td>24</td>
<td>8</td>
<td>22</td>
<td>31</td>
<td>27</td>
<td>48</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>18</td>
<td>19</td>
<td>35</td>
<td>39</td>
<td>42</td>
<td>70</td>
<td>14</td>
</tr>
<tr>
<td>Arkansas (AR)</td>
<td>35</td>
<td>40</td>
<td>37</td>
<td>20</td>
<td>28</td>
<td>39</td>
<td>29</td>
<td>39</td>
<td>13</td>
<td>18</td>
<td>21</td>
<td>30</td>
<td>21</td>
<td>44</td>
<td>46</td>
<td>43</td>
<td>44</td>
<td>31</td>
</tr>
<tr>
<td>Delaware (DE)</td>
<td>54</td>
<td>54</td>
<td>52</td>
<td>53</td>
<td>50</td>
<td>56</td>
<td>58</td>
<td>65</td>
<td>51</td>
<td>43</td>
<td>45</td>
<td>45</td>
<td>50</td>
<td>52</td>
<td>55</td>
<td>49</td>
<td>39</td>
<td>45</td>
</tr>
<tr>
<td>Florida (FL)</td>
<td>19</td>
<td>21</td>
<td>22</td>
<td>8</td>
<td>18</td>
<td>31</td>
<td>28</td>
<td>57</td>
<td>25</td>
<td>24</td>
<td>26</td>
<td>30</td>
<td>34</td>
<td>55</td>
<td>57</td>
<td>52</td>
<td>48</td>
<td>41</td>
</tr>
<tr>
<td>Georgia (GA)</td>
<td>29</td>
<td>18</td>
<td>31</td>
<td>4</td>
<td>7</td>
<td>29</td>
<td>18</td>
<td>43</td>
<td>8</td>
<td>13</td>
<td>15</td>
<td>18</td>
<td>18</td>
<td>30</td>
<td>33</td>
<td>37</td>
<td>54</td>
<td>30</td>
</tr>
<tr>
<td>Kentucky (KY)</td>
<td>49</td>
<td>47</td>
<td>48</td>
<td>25</td>
<td>47</td>
<td>49</td>
<td>49</td>
<td>59</td>
<td>40</td>
<td>40</td>
<td>42</td>
<td>45</td>
<td>41</td>
<td>50</td>
<td>54</td>
<td>54</td>
<td>36</td>
<td>44</td>
</tr>
<tr>
<td>Louisiana (LA)</td>
<td>21</td>
<td>10</td>
<td>12</td>
<td>5</td>
<td>7</td>
<td>31</td>
<td>20</td>
<td>24</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>19</td>
<td>17</td>
<td>47</td>
<td>53</td>
<td>29</td>
<td>57</td>
<td>23</td>
</tr>
<tr>
<td>Maryland (MD)</td>
<td>52</td>
<td>49</td>
<td>49</td>
<td>24</td>
<td>45</td>
<td>55</td>
<td>45</td>
<td>57</td>
<td>36</td>
<td>37</td>
<td>41</td>
<td>48</td>
<td>49</td>
<td>55</td>
<td>60</td>
<td>46</td>
<td>35</td>
<td>42</td>
</tr>
<tr>
<td>Mississippi (MS)</td>
<td>10</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>14</td>
<td>8</td>
<td>18</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>40</td>
<td>24</td>
<td>25</td>
<td>87</td>
<td>14</td>
</tr>
<tr>
<td>Missouri (MO)</td>
<td>46</td>
<td>50</td>
<td>49</td>
<td>30</td>
<td>47</td>
<td>55</td>
<td>50</td>
<td>56</td>
<td>35</td>
<td>38</td>
<td>48</td>
<td>48</td>
<td>42</td>
<td>51</td>
<td>50</td>
<td>50</td>
<td>36</td>
<td>45</td>
</tr>
<tr>
<td>North Car. (NC)</td>
<td>45</td>
<td>40</td>
<td>46</td>
<td>12</td>
<td>42</td>
<td>43</td>
<td>55</td>
<td>29</td>
<td>29</td>
<td>27</td>
<td>26</td>
<td>33</td>
<td>33</td>
<td>46</td>
<td>49</td>
<td>48</td>
<td>44</td>
<td>40</td>
</tr>
<tr>
<td>South Car. (SC)</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>49</td>
<td>25</td>
<td>49</td>
<td>59</td>
<td>39</td>
</tr>
<tr>
<td>Tennessee (TN)</td>
<td>45</td>
<td>43</td>
<td>46</td>
<td>24</td>
<td>43</td>
<td>51</td>
<td>44</td>
<td>54</td>
<td>32</td>
<td>31</td>
<td>33</td>
<td>39</td>
<td>37</td>
<td>50</td>
<td>49</td>
<td>53</td>
<td>44</td>
<td>38</td>
</tr>
<tr>
<td>Texas (TX)</td>
<td>31</td>
<td>22</td>
<td>22</td>
<td>9</td>
<td>17</td>
<td>24</td>
<td>20</td>
<td>52</td>
<td>11</td>
<td>12</td>
<td>19</td>
<td>17</td>
<td>25</td>
<td>53</td>
<td>55</td>
<td>49</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>Virginia (VA)</td>
<td>44</td>
<td>37</td>
<td>38</td>
<td>17</td>
<td>32</td>
<td>38</td>
<td>33</td>
<td>54</td>
<td>30</td>
<td>29</td>
<td>32</td>
<td>37</td>
<td>41</td>
<td>56</td>
<td>55</td>
<td>52</td>
<td>46</td>
<td>43</td>
</tr>
<tr>
<td>West Virginia (WV)</td>
<td>54</td>
<td>55</td>
<td>53</td>
<td>21</td>
<td>49</td>
<td>55</td>
<td>49</td>
<td>58</td>
<td>44</td>
<td>39</td>
<td>43</td>
<td>45</td>
<td>42</td>
<td>48</td>
<td>47</td>
<td>54</td>
<td>32</td>
<td>40</td>
</tr>
</tbody>
</table>
### 1. Republican Vote for President

<table>
<thead>
<tr>
<th>State</th>
<th>00</th>
<th>04</th>
<th>08</th>
<th>12</th>
<th>16</th>
<th>20</th>
<th>24</th>
<th>28</th>
<th>32</th>
<th>36</th>
<th>40</th>
<th>44</th>
<th>48</th>
<th>52</th>
<th>56</th>
<th>60</th>
<th>64</th>
<th>68</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alabama (AA)</td>
<td>35</td>
<td>21</td>
<td>24</td>
<td>8</td>
<td>22</td>
<td>31</td>
<td>27</td>
<td>48</td>
<td>14</td>
<td>13</td>
<td>14</td>
<td>18</td>
<td>19</td>
<td>35</td>
<td>39</td>
<td>42</td>
<td>70</td>
<td>14</td>
</tr>
<tr>
<td>Arkansas (AR)</td>
<td>35</td>
<td>40</td>
<td>37</td>
<td>20</td>
<td>28</td>
<td>39</td>
<td>29</td>
<td>39</td>
<td>13</td>
<td>18</td>
<td>21</td>
<td>30</td>
<td>21</td>
<td>44</td>
<td>46</td>
<td>43</td>
<td>44</td>
<td>31</td>
</tr>
<tr>
<td>Delaware (DE)</td>
<td>54</td>
<td>54</td>
<td>52</td>
<td>53</td>
<td>50</td>
<td>56</td>
<td>58</td>
<td>65</td>
<td>51</td>
<td>43</td>
<td>45</td>
<td>45</td>
<td>50</td>
<td>52</td>
<td>55</td>
<td>49</td>
<td>39</td>
<td>45</td>
</tr>
<tr>
<td>Florida (FL)</td>
<td>19</td>
<td>21</td>
<td>22</td>
<td>8</td>
<td>18</td>
<td>31</td>
<td>28</td>
<td>57</td>
<td>25</td>
<td>24</td>
<td>26</td>
<td>30</td>
<td>34</td>
<td>55</td>
<td>57</td>
<td>52</td>
<td>48</td>
<td>41</td>
</tr>
<tr>
<td>Georgia (GA)</td>
<td>29</td>
<td>18</td>
<td>31</td>
<td>4</td>
<td>7</td>
<td>29</td>
<td>18</td>
<td>43</td>
<td>8</td>
<td>13</td>
<td>15</td>
<td>18</td>
<td>18</td>
<td>30</td>
<td>33</td>
<td>37</td>
<td>54</td>
<td>30</td>
</tr>
<tr>
<td>Kentucky (KY)</td>
<td>49</td>
<td>47</td>
<td>48</td>
<td>25</td>
<td>47</td>
<td>49</td>
<td>49</td>
<td>59</td>
<td>40</td>
<td>40</td>
<td>42</td>
<td>45</td>
<td>41</td>
<td>50</td>
<td>54</td>
<td>54</td>
<td>36</td>
<td>44</td>
</tr>
<tr>
<td>Louisiana (LA)</td>
<td>31</td>
<td>10</td>
<td>12</td>
<td>5</td>
<td>7</td>
<td>31</td>
<td>20</td>
<td>24</td>
<td>7</td>
<td>11</td>
<td>14</td>
<td>19</td>
<td>17</td>
<td>47</td>
<td>53</td>
<td>29</td>
<td>57</td>
<td>23</td>
</tr>
<tr>
<td>Maryland (MD)</td>
<td>52</td>
<td>49</td>
<td>49</td>
<td>24</td>
<td>45</td>
<td>55</td>
<td>45</td>
<td>57</td>
<td>36</td>
<td>37</td>
<td>41</td>
<td>48</td>
<td>49</td>
<td>55</td>
<td>60</td>
<td>46</td>
<td>35</td>
<td>42</td>
</tr>
<tr>
<td>Mississippi (MI)</td>
<td>10</td>
<td>5</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>14</td>
<td>8</td>
<td>18</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>3</td>
<td>40</td>
<td>24</td>
<td>25</td>
<td>87</td>
<td>14</td>
</tr>
<tr>
<td>Missouri (MO)</td>
<td>46</td>
<td>50</td>
<td>49</td>
<td>30</td>
<td>47</td>
<td>55</td>
<td>50</td>
<td>56</td>
<td>35</td>
<td>38</td>
<td>48</td>
<td>48</td>
<td>42</td>
<td>51</td>
<td>50</td>
<td>50</td>
<td>36</td>
<td>45</td>
</tr>
<tr>
<td>North Car. (NC)</td>
<td>45</td>
<td>40</td>
<td>46</td>
<td>12</td>
<td>42</td>
<td>43</td>
<td>55</td>
<td>29</td>
<td>29</td>
<td>27</td>
<td>26</td>
<td>33</td>
<td>33</td>
<td>46</td>
<td>49</td>
<td>48</td>
<td>44</td>
<td>40</td>
</tr>
<tr>
<td>South Car. (SC)</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>49</td>
<td>25</td>
<td>49</td>
<td>59</td>
</tr>
<tr>
<td>Tennessee (TN)</td>
<td>45</td>
<td>43</td>
<td>46</td>
<td>24</td>
<td>43</td>
<td>51</td>
<td>44</td>
<td>54</td>
<td>32</td>
<td>31</td>
<td>33</td>
<td>39</td>
<td>37</td>
<td>50</td>
<td>49</td>
<td>53</td>
<td>44</td>
<td>38</td>
</tr>
<tr>
<td>Texas (TX)</td>
<td>31</td>
<td>22</td>
<td>22</td>
<td>9</td>
<td>17</td>
<td>24</td>
<td>20</td>
<td>52</td>
<td>11</td>
<td>12</td>
<td>19</td>
<td>17</td>
<td>25</td>
<td>53</td>
<td>55</td>
<td>49</td>
<td>37</td>
<td>40</td>
</tr>
<tr>
<td>Virginia (VA)</td>
<td>44</td>
<td>37</td>
<td>38</td>
<td>17</td>
<td>32</td>
<td>38</td>
<td>33</td>
<td>54</td>
<td>30</td>
<td>29</td>
<td>32</td>
<td>37</td>
<td>41</td>
<td>56</td>
<td>55</td>
<td>52</td>
<td>46</td>
<td>43</td>
</tr>
<tr>
<td>West Virginia (WV)</td>
<td>54</td>
<td>55</td>
<td>53</td>
<td>21</td>
<td>49</td>
<td>55</td>
<td>49</td>
<td>58</td>
<td>44</td>
<td>39</td>
<td>43</td>
<td>45</td>
<td>42</td>
<td>48</td>
<td>47</td>
<td>54</td>
<td>32</td>
<td>40</td>
</tr>
</tbody>
</table>
• Clustering:
  • Looking for similar rows
  • Low rank row sets

• Bi-Clustering:
  • Looking for rows and columns simultaneously
  • Low rank submatrices
### Application – Computational Biology

- Patients and Gene
- Cases and Controls

<table>
<thead>
<tr>
<th>Gene 1</th>
<th>Gene 2</th>
<th>Gene 3</th>
<th>\cdots</th>
<th>\cdots \cdots</th>
<th>Gene M</th>
</tr>
</thead>
<tbody>
<tr>
<td>Person 1</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>Person L</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>Patient 1</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>Patient 2</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
<tr>
<td>Patient N</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
<td>\cdots</td>
</tr>
</tbody>
</table>

\[
Z_{L \times M}
\]

\[
A_{N \times M}
\]
33268 gene expression measurements

$N_D = 193$ cases

$N_X = 128$ controls

ignored
Previous Attempts

• Naïve way:
  • check every possible submatrics
  • NP

• Iterative random methods:
  • Start from a initial submatrix and add/remove rows and columns to make submatrix low rank
  • A good Initial Condition is needed

• Spectral biclustering:
  • Iteratively remove rows and columns to get a low rank submatrix ---using SVD in every step
  • \( O(N^4) \times \) big constant
Our Algorithm

- For a given $N \times M$ matrix $A$, we are looking for some rows and columns whose intersection is a low rank matrix.
- We give every row and column a score then eliminate rows and columns with lower scores.

Let $A$ be a $n \times m \{1, -1\}$ matrix. Let $A_i$ be the $i$th row and $R_i$ be its score. Similarly we define $B_j$ as the $j$th column and $C_j$ as its scores.

Let $Z$ be a $d \times m \{1, -1\}$ matrix and $Z_i$ and $K_j$ be rows and columns.

Define $\langle \cdot, \cdot \rangle$ as the product of 2 vectors.

Then we can write SCORES as:

$$R_i = \sum_j < A_j, A_i >^2 - \sum_j < Z_j, A_i >^2$$  \hspace{1cm} (1)$$

$$C_i = \sum_j < B_j, B_i >^2 - \sum_j < B_j, B_i > < K_j, K_i >$$  \hspace{1cm} (2)$$
Algorithm Description

I. Pre-algorithm: Transform A into a binary matrix B

II. Main Algorithm
   1. Give a score to each rows and columns
   2. Remove rows and columns with lowest scores
   3. Recalculate the scores for all remaining rows and columns, go to step 2

III. The rows and columns remains at last forms a low rank submatrix.
Part 2: High Performance Computing
Complexity

- 2N iterations
- Calculate 2N scores in every step
- Recall Scores:

\[
R_i = \sum_j <A_j, A_i>^2 - \sum_j <Z_j, A_i>^2
\]

\[
C_i = \sum_j <B_j, B_i>^2 - \sum_j <B_j, B_i><K_j, K_i>
\]

\[
O(N^3) \quad \rightarrow \quad O(N^4)
\]
Contd.

• Luckily, we found a way to not recalculate the score, but update them
• When a row(col) is removed, there is only a small part in each score changes, we only need to remove these parts from scores.
• When a row is removed:

\[ R'_i = R_i - < A_i, a >^2 \]
\[ C'_i = C_i - m - 2 \sum_{A_k \neq a} < A_k, a > a_i a_{ki} + \sum_{Z_k} < Z_k, a > a_k z_{ki} \]

• When a column is removed:

\[ C'_i = C_i - < B_i, b > \left[ < B_i, b > - < K_i, k > \right] \]
\[ R'_i = R_i - n + d - 2 \sum_{B_k \neq b} b_i a_{ik} \left[ < B_k, b > - < K_k, k > \right] \]
• Luckily, we found a way to not recalculate the score, but update them.
• When a row (col) is removed, there is only a small part in each score changes, we only need to remove these parts from scores.
Complexity

• For a $n \times m$ matrix, the main body of our algorithm is:

1. Initialize scores
2. Remove according to scores
3. Update scores, and go to step 2

\[ O((n+m)nm) \]

\[ O(m) \text{ or } O(n) \]

\[ O(nm) \]

\[ O((n+m)nm) \]

\[ O((n+m)nm) \]

Memory bound
Problem Scale

- ASD (Sample data) \( 10^2 \times 10^4 \)
- Bipolar disorder \( 10^4 \times 10^6 \)
- Cancer \( 10^5 \times 10^7 \)
Memory Management

• Binary data can be compressed

• 32 entries:

  11101001000101011110101000100011

   ↓

   1 unsigned int

• Bipolar disorder $10^4 \times 10^6$ 1.1 GB
• Cancer $10^5 \times 10^7$ 110 GB

• 100GB RAM or read 100GB $10^7$ times or MPI
What makes our code fast!

• How to measure speed

\[ \text{Time Per Unit} = \frac{\text{running time/s}}{nm(n + m)} \]

\[ \text{Speedup} = \frac{\text{sequential time/s}}{\text{parallel time/s}} \]

• Test data:
  • ASD Sample data
  • Random matrix
What makes our code fast!

- Sequential version 0 MATLAB

SUPERSLOW
What makes our code fast!

• Sequential version 1

• ASD data:
  • Running time: 120s
  • TPU: 8.56239321e-9
What makes our code fast!

- Sequential version 2
- Basic optimization
- Improvement on Bitwise operation
  - Popcount
    - Counting bits in parallel
- ASD data:
  - Running time: 80s
  - TPU: 5.70826214e-9
What makes our code fast!

- Sequential version 3
- Loop structure optimization
  - Loop unrolling
    - For(i=0;i<16;i++)
      Operation[i]
      -> Operation[0];
      Operation[1];
    ......
      Operation[15];
- ASD data:
  - Running time: 38s
  - TPU: 2.71142452e-9
What makes our code fast!

- Sequential version 4
- Improvement in memory access
  - Use of cache
    Operation[0,array[s]];
    Operation[1,array[s]];
    ......
    Operation[15,array[s]];
    ->
    temp = array[s];
    Operation[0,temp];
    Operation[1,temp];
    ......
    Operation[15,temp];

- ASD data:
  - Running time: 27s
  - TPU: 1.92653847e-9
What makes our code fast!

• Sequential version 5
• Simply add “–O3”
• ASD data:
  • Running time: 17.8s
  • TPU: 1.27008833e-9
Contd.
What makes our code fast!

- OpenMP parallel version
- Choose the right loop to parallel
- Avoid synchronization

ASD data:
- 4-core
  - Running time: 6.01703s
  - TPU: 4.2933267e-10
  - Speedup: 2.9

- 12-core
  - Running time: 3.14378s
  - TPU: 2.2404929e-10
  - Speedup: 5.6
What makes our code fast!

- OpenMP version – core numbers

![Graphs showing performance of different number of threads for 640 * 12800 and 640 * 51200 matrices.](image)
What makes our code fast!

- OpenMP version – Large Data (random matrix)
- Different ratio of n and m
What makes our code fast!

- OpenMP version – Large Data (random matrix)
- Bipolar disorder \(10^4 \times 10^6\) 1 week
- Running time estimation

\[
TPU \times nm(n + m) = 1.2 \times 10^{-10} \times 10240 \times 819200 \times 829440
\]

\[= 834941s = 231h = 9d\]
What’s next…

• Run our code on Bipolar disorder data $10^4 \times 10^6$
• Generalization of the original bicluster algorithm !

• MPI version
• A way has been found
Even more ambitious...

• Possible MPI version
• Just found a way of avoiding too much data transfer between different computer.
  • Only $O(n)$ MPI_Reduction in every iteration
• Separately store data

• Run on $10^5 \times 10^7$ Cancer data Possible!
• Still very challenging
  • Ideally, with current efficiency, given 100 nodes each with 12 core. Estimated:
• Thank you.