Improving $k$-means Clustering with Genetic Programming for Feature Construction

Andrew Lensen, Bing Xue, and Mengjie Zhang
Victoria University of Wellington - New Zealand

andrew.lensen@ecs.vuw.ac.nz

Clustering and Feature Construction

- **Clustering**: grouping related instances into $K$ clusters.
- **$k$-means** is the most commonly used clustering algorithm, but has fundamental limitations:
  - Scales poorly to large dimensionality.
  - Struggles with many clusters (high $K$).
  - Very dependent on initial random centroids.
- Can improve $k$-means by using fewer, more-powerful features to partition the data more accurately:
  - Use feature selection and construction.

Existing Methods

- Handful of existing work using GP for clustering, but none performing explicit feature construction to improve the performance of a clustering algorithm.

Goal

Propose new GP representations and fitness functions to automatically select and construct multiple features to improve the performance of $k$-means.

- Using a wrapper approach, where the features produced are fed to $k$-means for clustering.

Representation #1: Multi-Tree GP

- Use multiple trees, each of which produces a single constructed feature as the tree output.
- Produce $t$ constructed features for $t$ trees.
- Terminals: feature set, random double values in $[0,1]$.
- Functions: several arithmetic operators, max/min/it.
- **Fig 1** shows an example of this representation, with a range of trees performing selection, and varying levels of feature construction.

Representation #2: Vector GP

- Use a single tree, which produces multiple constructed features as the tree output.
- A tree builds up a vector of constructed features.
- Produce a variable number of constructed features.
- Extend the above function set to operate on two vectors in a pair-wise manner.
- Add a new *concat* function which concatenates two vector inputs into one vector output.
- **Fig 2** shows an example of the vector representation, which selects and constructs several features.

Fitness Function

- Test how the performance of $k$-means is improved when using different fitness functions:
  - **Total Intra-Variance**: the sum of distance from each instance to its cluster mean. This is what $k$-means is designed to optimise.
  - **Connectedness**: How well each instance is in the same cluster as its nearest neighbours. Similar instances should belong to the same cluster.
- Can train $k$-means with many different functions!

Experiments & Results

- Compared each of the two representations and two fitness functions against $k$-means (with All Features) across a range of synthetic datasets.
- 50d10c → 50 features, 10 clusters.
- Measured the F-measure – how well the clusters produced match the known cluster labels.
- +/- indicate significant improvement/deterioration at 95% CI over 30 runs vs original $k$-means.

<table>
<thead>
<tr>
<th>Method</th>
<th>50d10c</th>
<th>50d20c</th>
<th>50d40c</th>
<th>100d10c</th>
<th>100d20c</th>
<th>100d40c</th>
</tr>
</thead>
<tbody>
<tr>
<td>MTConn</td>
<td>0.5167+</td>
<td>0.4996+</td>
<td>0.4397+</td>
<td>0.5311</td>
<td>0.4657+</td>
<td>0.4629+</td>
</tr>
<tr>
<td>MTIntra</td>
<td>0.4785+</td>
<td>0.4776+</td>
<td>0.4269+</td>
<td>0.5825+</td>
<td>0.4598+</td>
<td>0.462+</td>
</tr>
<tr>
<td>VectorConn</td>
<td>0.5005</td>
<td>0.4832+</td>
<td>0.4106+</td>
<td>0.5446</td>
<td>0.4451+</td>
<td>0.4418+</td>
</tr>
<tr>
<td>VectorIntra</td>
<td>0.4795+</td>
<td>0.4351+</td>
<td>0.3759+</td>
<td>0.5854+</td>
<td>0.4331+</td>
<td>0.4028+</td>
</tr>
<tr>
<td>$k$-means AF</td>
<td>0.4939</td>
<td>0.3823</td>
<td>0.2618</td>
<td>0.5255</td>
<td>0.3800</td>
<td>0.2675</td>
</tr>
</tbody>
</table>

- GP shows significant improvement on 79% of results.
- **Connectedness** generally outperforms Total Intra.
- **Multi-tree** generally outperforms Vector.
- GP has highest improvement when $K$ is large.

Future Work

- Further investigating new fitness functions to further improve performance.
- Apply this approach when $K$ is unknown.
- Automatically determine the number of trees, $t$.

Figure 1: An example program on the 100d20c dataset with F-measure of 0.9947 using the multi-tree approach.

Figure 3: An example program on the 100d60c dataset with F-measure of 0.899 using the vector approach.