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

Andrew Lensen, Bing Xue, and Mengjie Zhang

VICTORIA UNIVERSITY OF WELLINGTON
ECRG
Evolutionary Computation Research Group

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.

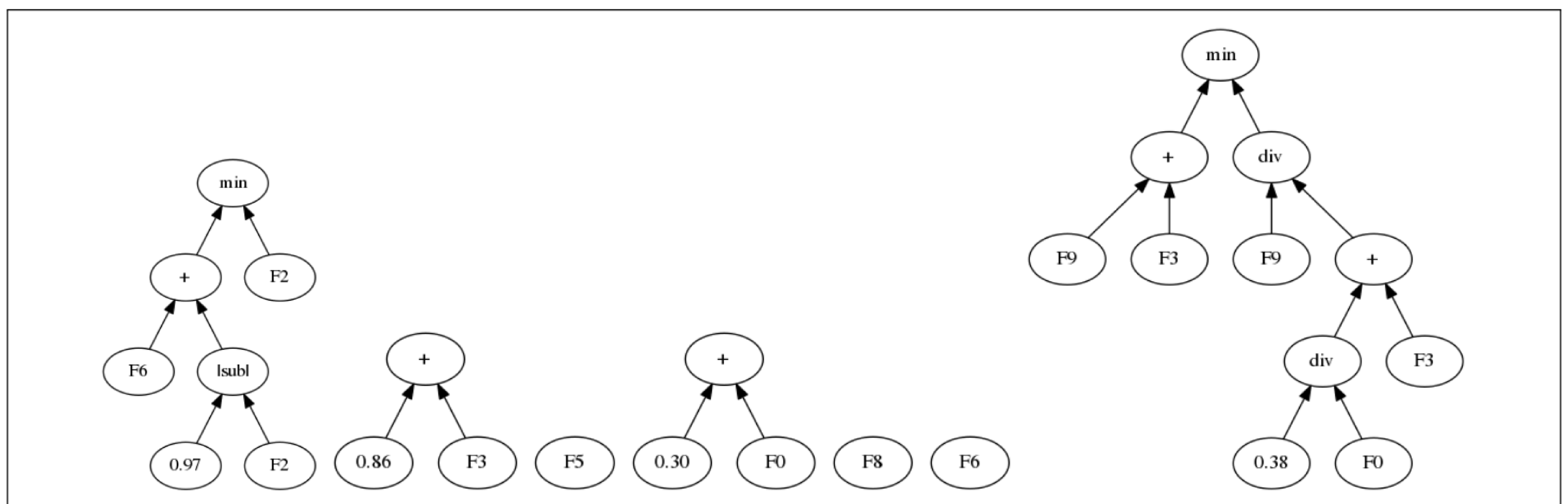
- Using a **wrapper** approach, where the features produced are fed to *k*-means for clustering.

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/if.
- **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.



(a) The evolved multi-tree GP individual with 7 trees.

$$\left[min(F_6 + |0.97 - F_2|, F_2),\quad (0.86 + F_3),\quad F_5,\quad (0.3 + F_0),\quad F_8,\quad F_6,\quad min\left((F_9 + F_3), \frac{F9}{(\frac{0.38}{F_0} + F_3)}\right)\right]$$

(b) Constructed feature set generated by the programs.

Figure 1: An example program on the 10d20c dataset with F-measure of 0.9947 using the *multi-tree* approach.

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 **A**ll **F**eatures) 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.

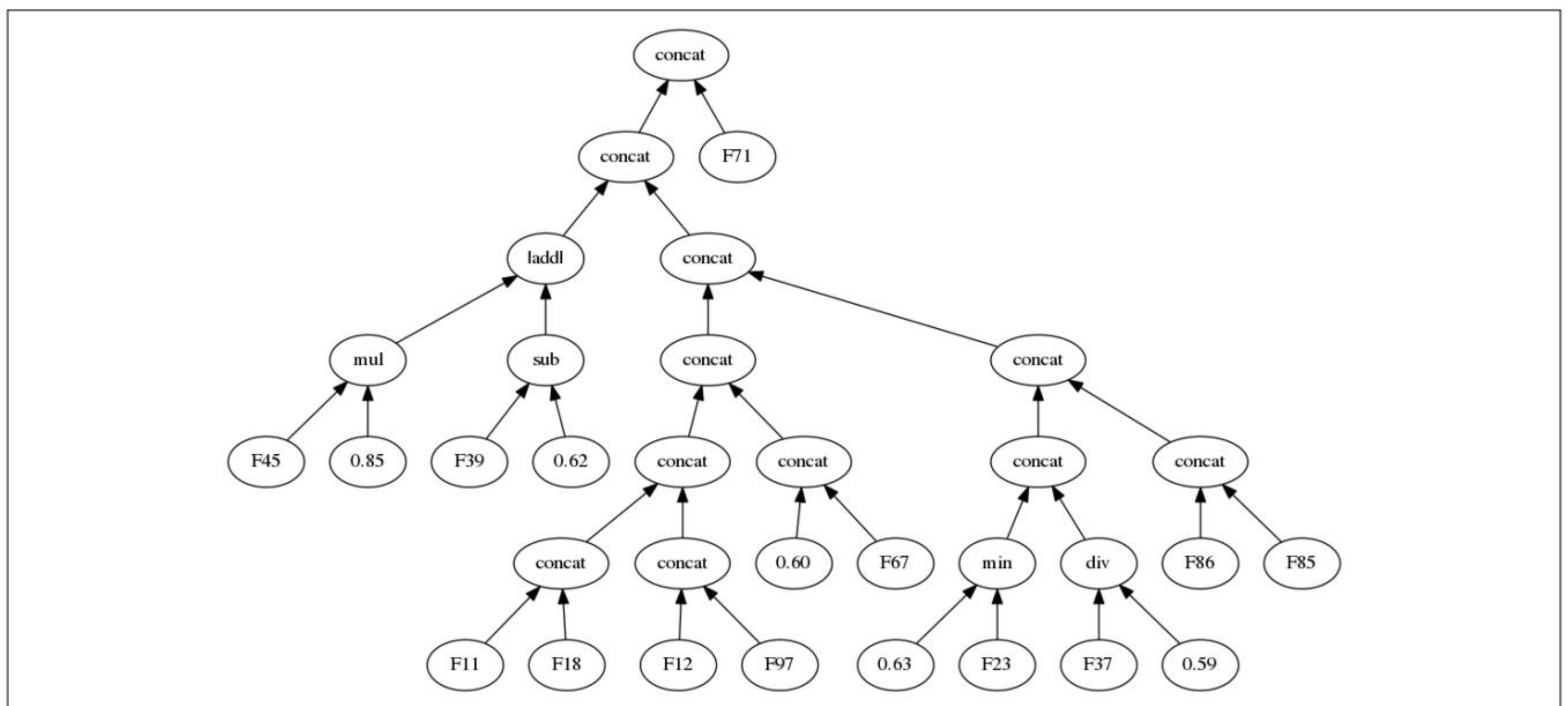
Table 1: F-measure performance on the datasets

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

- 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***.



(a) The GP tree.

$$\left[|(F_{45} \times 0.85) + (F_{39} - 0.62)|,\quad F_{11},\quad F_{18},\quad F_{12},\quad F_{97},\quad 0.6,\quad F_{67},\quad min(0.63, F_{23}),\quad \frac{F_{37}}{0.59},\quad F_{86},\quad F_{85},\quad F_{71}\right]$$

(b) Constructed feature set generated by the program.

Figure 2: An example program on the 100d40c dataset with F-measure of 0.499 using the *vector* approach.