Learning Program Structure and Assigning Style Grade by k-means clustering and Softmax Regression

Abstract:

The goal of this research is to figure out how to automatically assign a style grade to a program and provide style feedback. More specifically, the procedure employs Karel Programs from the first assignment of the CS106A class.

Motivation:

While identifying a functionality mistake can be straightforward, identifying style mistakes is subject to subtle conventions and an infinite number of possible programs. This poses a problem for students trying to get feedback on their program style while they are still working on the assignment.

Approach:

- ➢ Cluster the training data of working programs into **k means**
- ➢ Explore three clustering strategies and choose one
- ➢ Create a **Logistic Regression** or Naive Bayes model of the functions of the programs in each cluster including non-working programs and decide for each function of a test program whether the function is well decomposed and whether is well formatted. Use this to provide feedback to the student.
- ➢ Then average these two parameters for all functions in a program and use them to describe the program.
- ➢ Finally add whether the program works to the program vector and use this vectors to run **Softmax Regression** on the test program and decide its style grade.

Implementation:

Strategy 1

100 programs from the same assignment

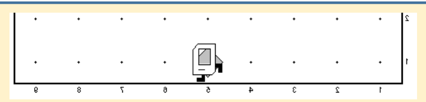
```
public class MidpointFindingKarel
    public void run(){
        placeBeepers();
        pickBeeper();
        returnToInitialPosition();
        if(beepersPresent()){
            pickBeeper();
            moveForward();
        }
        markMidpoint();

    }
```

Parse into primitives
And create a vector with counts of each primitive

[#move(); #putBeeper(); #pickBeeper(); #turnLeft();#turnRight();#turnAround()]

Strategy 2



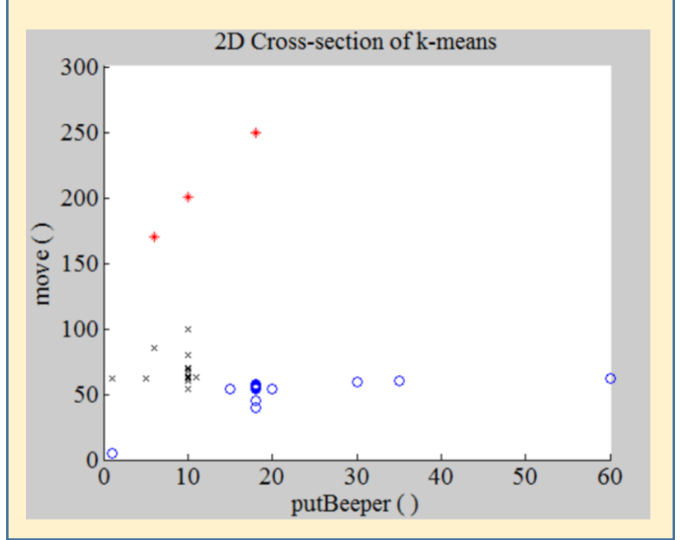
Keep track of Karel's Coordinates over time

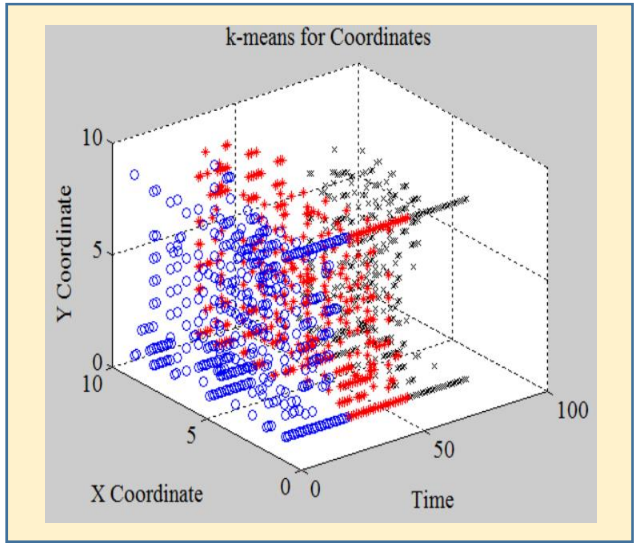
[time,x,y; time2,x2,y2; …]

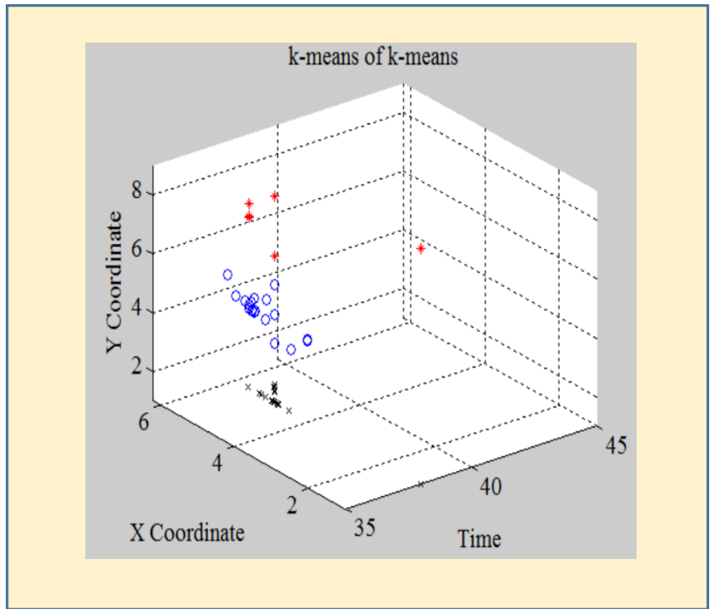
Strategy 3

Run preliminary k-means to find cluster of a single program

Cluster into k-means







Clustering results:

- ➢ Clustering by counts of primitives is very susceptible to unnecessary command calls. Also it does not distinguish between call times.
- ➢ Clustering by pouring all coordinates may be susceptible to outlying programs that tend to spend too much time at a certain place.
- ➢ Double k-means overall seems to address the issues above and can even be implemented for variable times. However, it breaks down with infinite loops.

Logistic Regression or Naïve Bayes to decide whether a program's function is well decomposed and well formatted.

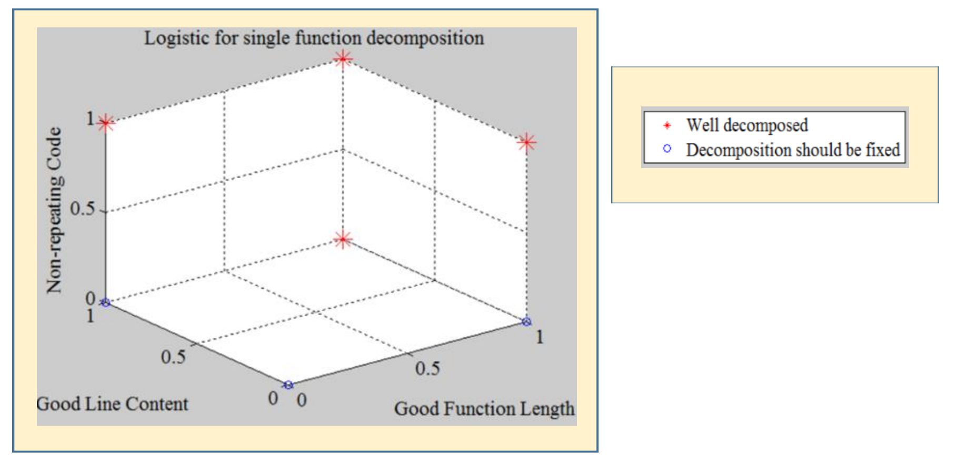
Fit the functions in the programs in each cluster using collected properties of each function

[between 2-10 lines?; Appropriate line length; No repeating code; label]

[Right indentation; No blank lines; Commented]

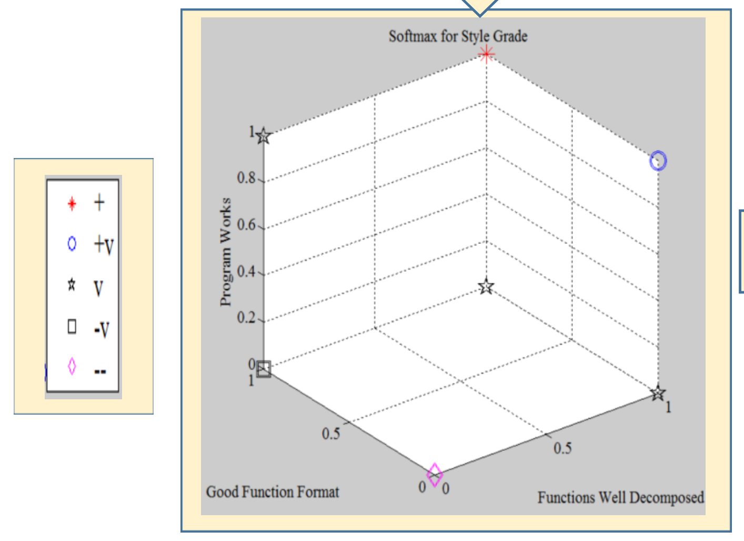
Is the function well decomposed?

Is the function well formatted?



Then, take average over a program.
[Program decomposed; program well formatted; Program Works]

Fit Softmax Model



Report Style grades for the program overall and each function. Provide feedback on how to improve the style of each function.

Softmax results:

- ➢ Softmax with clustering performed with 87% accuracy while Softmax without clustering performed at about 75%. Therefore, clustering before hand seems to perform better.

Future Work:

- ➢ Apply these results into a purely Java setting
- ➢ Account for variable naming and comment content.

Homero Roman Roman
CS229 2016: Final Project
Stanford University