

CMDA 3634 Spring 2022: Fuzzy k-means project part 1

Jonathan Baker & Tim Warburton

April 2022

In this assignment you will begin working on a fuzzy k-means algorithm. You will write code to measure distances between points and calculate weighted means.

You don't need to do anything to support parallelism for this assignment (no OpenMP, no MPI).

1 k-means and fuzzy Lloyd's algorithm

If you'd like an alternative introduction to fuzzy k-means, there is one here:

https://en.wikipedia.org/wiki/Fuzzy_clustering#Fuzzy_C-means_clustering

Euclidean distance between two vectors $\mathbf{v}_0 = (x_0, y_0)$ and $\mathbf{v}_1 = (x_1, y_1)$ is defined by

$$\|\mathbf{v}_0 - \mathbf{v}_1\| = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}. \quad (1)$$

Given cluster centers, $\mathbf{c}_1, \dots, \mathbf{c}_k$, and a real fuzziness parameter $m > 1$, the “weight” function (or dissimilarity) between a vector \mathbf{v}_i and a cluster center \mathbf{c}_j is

$$w_{i,j} = \left(\sum_{\ell=0}^{k-1} \left(\frac{\|\mathbf{v}_i - \mathbf{c}_j\|^2}{\|\mathbf{v}_i - \mathbf{c}_\ell\|^2} \right)^{1/(m-1)} \right)^{-1}. \quad (2)$$

Given data points $\mathbf{v}_0, \dots, \mathbf{v}_{n-1}$, the weighted mean of the data with respect to cluster center \mathbf{c}_j is

$$\hat{\mathbf{c}}_j = \frac{\sum_{i=0}^{n-1} \mathbf{v}_i w_{i,j}^m}{\sum_{i=0}^{n-1} w_{i,j}^m}. \quad (3)$$

The superscripts m in (3) are exponents: each $w_{i,j}$ is to be raised to the m th power. **To keep things simple, assume $m = 2$ until further notice.**

Fuzzy Lloyd's algorithm repeatedly updates the cluster centers with the weighted mean of the data with respect to the current cluster center. In other words, it repeatedly replaces \mathbf{c}_j with $\hat{\mathbf{c}}_j$.

1.1 Example

Here is some example data

$$\mathbf{v}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \quad \mathbf{v}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}.$$

Here are example cluster centers

$$\mathbf{c}_0 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \mathbf{c}_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

The distances between \mathbf{c}_0 and the vectors are

$$\|\mathbf{v}_0 - \mathbf{c}_0\| = 1, \quad \|\mathbf{v}_1 - \mathbf{c}_0\| = \sqrt{2}, \quad \|\mathbf{v}_2 - \mathbf{c}_0\| = \frac{\sqrt{17}}{2}.$$

The distances between \mathbf{c}_1 and the vectors are

$$\|\mathbf{v}_0 - \mathbf{c}_1\| = \sqrt{5}, \quad \|\mathbf{v}_1 - \mathbf{c}_1\| = 2, \quad \|\mathbf{v}_2 - \mathbf{c}_1\| = \frac{\sqrt{5}}{2}.$$

The weights (dissimilarities) between the data and the cluster centers are

$$\begin{aligned} w_{0,0} &= \left(\frac{1^2}{1^2} + \frac{1^2}{\sqrt{5}^2} \right)^{-1} = \frac{5}{6}, \\ w_{0,1} &= \left(\frac{\sqrt{5}^2}{1^2} + \frac{\sqrt{5}^2}{\sqrt{5}^2} \right)^{-1} = \frac{1}{6}, \\ w_{1,0} &= \left(\frac{\sqrt{2}^2}{\sqrt{2}^2} + \frac{\sqrt{2}^2}{2^2} \right)^{-1} = \frac{2}{3}, \\ w_{1,1} &= \left(\frac{2^2}{\sqrt{2}^2} + \frac{2^2}{2^2} \right)^{-1} = \frac{1}{3}, \\ w_{2,0} &= \left(\frac{(\sqrt{17}/2)^2}{(\sqrt{17}/2)^2} + \frac{(\sqrt{17}/2)^2}{(\sqrt{5}/2)^2} \right)^{-1} = \frac{5}{22}, \\ w_{2,1} &= \left(\frac{(\sqrt{5}/2)^2}{(\sqrt{17}/2)^2} + \frac{(\sqrt{5}/2)^2}{(\sqrt{5}/2)^2} \right)^{-1} = \frac{17}{22}. \end{aligned}$$

The weighted mean of the data with respect to \mathbf{c}_0 is

$$\hat{\mathbf{c}}_0 = \frac{\left(\frac{5}{6}\right)^2 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \left(\frac{2}{3}\right)^2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \left(\frac{5}{22}\right)^2 \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}}{\left(\frac{5}{6}\right)^2 + \left(\frac{2}{3}\right)^2 + \left(\frac{5}{22}\right)^2} \approx \begin{bmatrix} 0.0434 \\ 0.3950 \end{bmatrix}.$$

The weighted mean of the data with respect to \mathbf{c}_1 is

$$\hat{\mathbf{c}}_1 = \frac{\left(\frac{1}{6}\right)^2 \begin{bmatrix} 0 \\ 0 \end{bmatrix} + \left(\frac{1}{3}\right)^2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \left(\frac{17}{22}\right)^2 \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}}{\left(\frac{1}{6}\right)^2 + \left(\frac{1}{3}\right)^2 + \left(\frac{17}{22}\right)^2} \approx \begin{bmatrix} 0.8113 \\ 0.5566 \end{bmatrix}.$$

If this example were given to your **fuzzykmeans** function, it should overwrite the contents of the **centers** array with the coordinates of $\hat{\mathbf{c}}_0$ and $\hat{\mathbf{c}}_1$.

2 Implementation

2.1 Vector arrays

For this assignment, you'll be representing collections of vectors in R^2 as double arrays, `double*`. Since each vector has two dimensions (x and y), the arrays will be twice as long as the number of vectors being represented. The array entries of the i th vector will be stored in the entries $2i$ and $(2i+1)$ of the array. This is sometimes called a “flattened” array. For example, the three data vectors in Section 1.1 would be represented in your program by an array with six entries,

$$\underbrace{\{0, 0\}}_{\mathbf{v}_0}, \underbrace{\{0, 1\}}_{\mathbf{v}_1}, \underbrace{\{1, 0.5\}}_{\mathbf{v}_2},$$

and the two cluster centers in the example would be represented in your program by an array with four entries,

$$\underbrace{\{-1, 0\}}_{\mathbf{c}_0}, \underbrace{\{2, 1\}}_{\mathbf{c}_1}.$$

2.2 The files

Instructors will push one file to your remote class Git repo:

- A header file with declarations of the functions listed above
 - This file is `fuzzylloyd/fuzzykmeans.h`
 - You don't have to edit this file, but if you write any extra supporting functions in `fuzzylloyd/fuzzykmeans.c`, they should be declared in this header file. You may want to add a `vec` constructor, for example.

You only have to write one new file:

- A C file where you implement the functions that are in declared in `fuzzylloydfuzzykmeans.h`.
 - Name this file `fuzzylloyd/fuzzykmeans.c`
 - You used a header file in HW 6, so you may want to look at the HW 6 project for a refresher on using a header file.

2.3 The functions

In your C file, you need to implement

- Euclidean distance `squared` (1)
 - `double dist2(double* v0, double* v1)`

- `v0` and `v1` are pointers to arrays of length 2, each representing 2-dimensional vectors
- `v0` and `v1` may be pointers into the middle of an array containing multiple vectors, but you can think of each of them as an array of length 2 representing a single vector each: `v0[0]` is the x coordinate of the first vector and so on.
- You never actually need distance itself, you only need distance squared—which is easier to calculate anyway.
- The weight function (2) calculating $w_{i,j}$
 - `double weight(double* vi, double* centers, int j, int k)`
 - `vi` is a pointer to a single data vector representing the i th data vector \mathbf{v}_i
 - `centers` is a pointer to all of the cluster center vectors. Notice that you will need the distance from \mathbf{v}_i to **all** of the cluster centers in order to calculate the weight $w_{i,j}$, even though $w_{i,j}$ is the weight associated with just one cluster center.
 - `j` specifies the cluster in which the weight of \mathbf{v}_i is being calculated
 - `k` is the number of clusters (so the array `centers` is has $2k$ entries)
- A fuzzy Lloyd update using (3) to update $\mathbf{c}_j \leftarrow \hat{\mathbf{c}}_j$ for each j .
 - `void fuzzykmeans(double* data, int n, double* centers, int k)`
 - `data` is a pointer to all of the data vectors
 - `n` is the number of data points (so the array `data` is has $2n$ entries)
 - `centers` is a pointer to all of the cluster center vectors
 - `k` is the number of clusters (so the array `centers` is has $2k$ entries)

3 Testing and expectations

You may compose and run any tests that you want, **but if you write a `main` function, it must *not* be in `fuzzylloyd/fuzzykmeans.c`**. If you put a `main` in `fuzzymeans.c`, it will interfere with my own `main` function when I compile your code into a library file for my tests.

- Your code should compile without producing warnings or errors from the compiler.
- Each of your functions should have the correct behavior.
 - Most of your functions need to return the correct value, but instead of returning a value, `fuzzykmeans` should use pointers to overwrite the array `centers`.
- Your functions should not produce memory errors or leaks.
 - You must not read any uninitialized memory or write to any invalid or out-of-bounds addresses.
 - If your functions allocate heap memory, then they should **free** that memory.

4 Submission

Push your source code `fuzzylloyd/fuzzykmeans.c` to your remote class Git repository. If you changed `fuzzylloyd/fuzzykmeans.h`, then push that too.

You are *not required* to submit any main function files, makefiles, Slurm submission files, or output files. But for your own benefit, you *should* write a main function to test your code and a makefile to simplify compilation.