Лабораторная работа №1 – Кластеризация методом нечетких с-средних (fuzzy c-means)

Теоретическая часть

Неформально кластеризацию можно описать как разделение множества объектов на подмножества (кластеры) таким образом, чтобы объекты одной группы были похожи между собой в большей степени, чем похожи на объекты из других групп.

Одним из самых известных методов кластеризации оптимизационного типа является метод k-средних (k-means), расширением которого является fuzzy c-means (FCM). Отличие FCM-алгоритма от k-means заключается в том, что вместо однозначной принадлежности объекта к тому или иному кластеру, FCM алгоритм возвращает для каждого объекта набор значений из диапазона $[0,1]$ которые определяют степень принадлежности объекта к каждому из найденных кластеров.

Суть алгоритма заключается в оптимизации функционала вида:

$$\sum_{l=1}^{c}\sum_{i=1}^{n}(\mu_{il})^m d(X_i, V_l) \to min,$$

где X_i – один из кластеризуемых объектов;
V_l – центр l-го кластера;
μ_{il} – степень принадлежности i-го объекта к l-му кластеру.
d – функция расстояния между объектами;
m – экспоненциальный вес ($m \geq 1$), обычно выбирают $m = 2$.

Каждый объект X_i описывается числовым вектором признаков размера m, т.е.

$$X_i = [x_1^i, \dots, x_m^i]$$

В качестве расстояний могут быть использованы, например, расстояние Хэмминга

$$h(X_i, X_j) = \sum_{k=1}^{m} |x_k^i - x_k^j|$$

и расстояние Евклида:

$$e(X_i, X_j) = \sqrt{\sum_{k=1}^{m} (x_k^i - x_k^j)}$$

Алгоритм FCM описывается следующей последовательностью шагов:

1) В качестве входного значения алгоритма задаются с – количество кластеров и ϵ – желаемая точность вычисления, а так же n векторов некоторого размера, описывающих численные признаки всех объектов
2) Генерируется случайным образом матрица $U^0 = (u_{il})_{n \times c}$ размера $n \times c$ значений принадлежности объектов к кластерам, таким образом, чтобы сумма по столбцам строки (сумма принадлежностей к разным кластерам одного объекта) была равна 1, т.е.:

$$\forall i \sum_{l=1}^{c} u_{il} = 1$$

3) Далее шаги выполняются итеративно. На k-ой итерации:
   - На основе матрицы значений принадлежности U^k находятся новые c центры классов по формуле:

$$V_l = \frac{1}{\sum_{i=1}^{n} \mu_{il}} \sum_{i=1}^{n} (\mu_{il})^m X_i$$

   - Рассчитывается матрица значений принадлежности U^{k+1} по формуле:

$$\mu_{ik} = \left(\sum_{j=1}^{c} \left(\frac{d(X_i, V_k)}{d(X_i, V_j)} \right)^{\frac{2}{m-1}} \right)^{-1}$$

   - Если выполняется условие

$$\left\| U^{k+1} - U^k \right\| \leq \epsilon$$

     где в качестве матричной нормы можно взять

$$\max_{i,l} \left| \mu_{il}^{k+1} - \mu_{il}^{k} \right|$$

     то вычисление оканчивается и U^{k+1} считается результатом кластеризации, иначе выполняется новая итерация.

Алгоритм можно начать со случайного выбора центров кластеров, вместо случайной инициализации матрицы принадлежностей. Тогда на следующем шаге происходит вычисление матрицы принадлежностей, а за тем происходят итерационные вычисления.

Задание

1. В результате выполнения лабораторной работы должно получится консольное приложение, кластеризующее переданные данные методом FCM
2. Реализовать парсинг csv файла. Обеспечить возможность настройки параметров формата файла – возможность выбирать разделители колонки, игнорировать первую строку (заголовок), первую колонку (там может быть порядковый номер объекта) и последнюю колонку (там может быть метка класса объекта).
3. Реализовать кластеризации методом FCM. Файл данных, количество кластеров и точность задаются пользователем. Так же пользователем задается в виде параметра используемая метрика. Программа должна предоставлять возможно использовать по крайней мере две метрики – Хэмминга и Евклида. Так же пользователь должен иметь возможность выбрать с чего начинается работа алгоритма – случайной инициализации матрицы принадлежностей или случайного выбора центров классов.
4. В качестве параметра должна быть возможность задать файл, в который должны быть записаны результаты кластеризации. Если он не задан – результат выводится на консоль.

5. В качестве результата выводится матрица принадлежностей.
6. Простые функции, вроде матричной нормы и расстояний должны быть покрыты юнит-тестами (минимум – 3 функции, покрытие тестами должно учитывать граничные случаи).
7. Программа должна обрабатывать возможные ошибки связанные с вводом-выводом (см. IOException) и выводить соответствующие сообщения.
8. Параметры командной строки должны быть описаны в файле readme. Там же должны быть описаны любые особенности компиляции, даже если программа собирается при помощи cabal.
9. Результат выполнения работы (исходники) должен быть загружен на GitHub, а ссылка на него прислана на stasshiray@gmail.com с темой вида группа – ФИО – номер работы.

При написании лабораторно работы можно использовать любые библиотеки с Hackage, если они не реализуют алгоритм FCM.

К условию прилагаются 3 файла с исходными данными. В каждом из них последняя колонка – метка класса.