ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГОБУ ВПО “СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ”

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

“Структуры и алгоритмы обработки данных”

на тему

Алгоритмы кластеризации. Алгоритм *k*-средних (*k-means*)

Выполнил студент *Тимошкин Владислав Николаевич*

Ф.И.О.

Группы *ИС – 341*

Работу принял _____ доцент к.т.н. М.Г. Курносов

подпись

Защищена _____ Оценка _____

Новосибирск – 2014

Оглавление

ВВЕДЕНИЕ

В современном мире информация имеет огромную ценность, поэтому важно уметь грамотно ее структурировать, обобщать и представлять для последующего анализа. С ростом объемов информации становится важным поиск оптимальных методов ее обработки.

Структуризация информации может потребоваться для поиска и сравнения похожих наборов данных, поиска дубликатов, составления списка данных смежной тематики или просто списка рекомендуемых данных. Решением этих задач занимается кластерный анализ. Целью данной работы является изучение алгоритмов кластеризации, в частности алгоритма k-means.

Кластеризацию, по большому счёту, можно рассматривать как обобщение сортировки. Сортируя список объектов, мы, по сути, выстраиваем все объекты в один ряд и с помощью компаратора (специального устройства для сравнения) выбираем в этом ряду первый объект, второй и так далее. В результате этого процесса для каждого объекта ряда мы идентифицируем его ближайших соседей.

Выполняя кластеризацию набора объектов, мы также идентифицируем ближайшее окружение объекта, но объекты при этом могут сохранять свою многомерную природу. Объектами могли бы быть точки на плоскости или в трехмерном пространстве, или точки более общего геометрического построения, в зависимости от числа учитываемых атрибутов и от понятия расстояния, которым мы хотим воспользоваться.

Задача алгоритмов кластеризации - идентифицировать группы способом, который, следовательно, можно распространить на несколько измерений и на пространство произвольных объектов. Члены кластера должны быть очень похожи друг на друга (на своих соседей) и иметь существенные отличия от членов любого другого кластера этого исходного набора.

1. Кластеризация. Виды алгоритмов

1.1. Определение кластеризации

Кластеризация (или **кластерный анализ)** — это задача разбиения множества объектов на группы, называемые кластерами. Внутри каждой группы должны оказаться «похожие» объекты, а объекты разных группы должны быть как можно больше отличными друг от друга. При кластеризации перечень групп четко не задан и определяется в процессе работы алгоритма (в отличие от классификации).

Применение кластерного анализа в общем виде сводится к следующим этапам:

1. Отбор выборки объектов для кластеризации.
2. Определение множества переменных, по которым будут оцениваться объекты в выборке. При необходимости - нормализация значений переменных.
3. Вычисление значений меры сходства между объектами.
4. Применение метода кластерного анализа для создания групп сходных объектов (кластеров).
5. Представление результатов анализа.

После получения и анализа результатов возможна корректировка выбранной метрики и метода кластеризации до получения оптимального результата.

Можно выделить две основные классификации алгоритмов кластеризации:

1. По способу обработки данных: иерархические и неиерархические.
2. По способу анализа данных: четкие и нечеткие.

1.2. Меры расстояний

Для того чтобы сравнивать два объекта, необходимо иметь критерий, на основании которого будет происходить сравнение. Как правило, таким критерием является расстояние между объектами.

1. Евклидово расстояние:

Наиболее распространенная функция расстояния. Представляет собой геометрическое расстояние в многомерном пространстве.

2. Квадрат евклидова расстояния:

Применяется для придания большего веса более отдаленным друг от друга объектам.

3. Расстояние городских кварталов (манхэттенское расстояние):

Это расстояние является средним разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако для этой меры влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат).

4. Расстояние Чебышева:

Это расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате.

5. Степенное расстояние:

Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются.

Выбор метрики полностью лежит на исследователе, поскольку результаты кластеризации могут существенно отличаться при использовании разных мер

1.3. Иерархические и неиерархические алгоритмы кластеризации

Иерархическая кластеризация (также **графовые алгоритмы кластеризации**) — совокупность алгоритмов упорядочивания данных, визуализация которых обеспечивается с помощью графов.

Алгоритмы упорядочивания данных указанного типа исходят из того, что некое множество объектов характеризуется определённой степенью связности. Предполагается наличие вложенных групп (кластеров различного порядка).

В случае использования иерархических алгоритмов встает вопрос, как объединять между собой кластера, как вычислять «расстояния» между ними. Существует несколько метрик:

• Одиночная связь (расстояния ближайшего соседа) В этом методе расстояние между двумя кластерами определяется расстоянием между двумя наиболее близкими объектами (ближайшими соседями) в различных кластерах.

• Полная связь (расстояние наиболее удаленных соседей). В этом методе расстояния между кластерами определяются наибольшим расстоянием между любыми двумя объектами в различных кластерах (т.е. наиболее удаленными соседями). Этот метод обычно работает очень хорошо, когда объекты происходят из отдельных групп. Если же кластеры имеют удлиненную форму или их естественный тип является «цепочечным», то этот метод непригоден.

• Невзвешенное попарное среднее. В этом методе расстояние между двумя различными кластерами вычисляется как среднее расстояние между всеми парами объектов в них. Метод эффективен, когда объекты формируют различные группы, однако он работает одинаково хорошо и в случаях протяженных («цепочечного» типа) кластеров.

• Взвешенное попарное среднее. Метод идентичен методу невзвешенного попарного среднего, за исключением того, что при вычислениях размер соответствующих кластеров (т.е. число объектов, содержащихся в них) используется в качестве весового коэффициента. Поэтому данный метод должен быть использован, когда предполагаются неравные размеры кластеров.

• Невзвешенный центроидный метод. В этом методе расстояние между двумя кластерами определяется как расстояние между их центрами тяжести.

• Взвешенный центроидный метод (медиана). Этот метод идентичен предыдущему, за исключением того, что при вычислениях используются веса для учета разницы между размерами кластеров. Поэтому, если имеются или подозреваются значительные отличия в размерах кластеров, этот метод оказывается предпочтительнее предыдущего.

Неиерархические алгоритмы основаны на оптимизации некоторой целевой функции, определяющей оптимальное в определенном смысле разбиение множества объектов на кластеры.

К неиерархическим алгоритмам кластеризации относятся алгоритмы k-means (а также его вариации - k-means++ и k-medians).

1.4. Четкие и нечеткие алгоритмы кластеризации

Существует множество методов кластеризации, которые можно классифицировать на четкие и нечеткие.

Четкие методы кластеризации разбивают исходное множество объектов X на несколько непересекающихся подмножеств. При этом любой объект из X принадлежит только одному кластеру.

Нечеткие методы кластеризации позволяют одному и тому же объекту принадлежать одновременно нескольким (или даже всем) кластерам, но с различной степенью. Нечеткая кластеризация во многих ситуациях более "естественна", чем четкая, например, для объектов, расположенных на границе кластеров.

2. Алгоритм кластеризации *k*-средних (*k-means*)

2.1. Происхождение и основные понятия алгоритма *k-means*

k-means наиболее популярный метод кластеризации. Был изобретён в 1950х годах математиком Гуго Штейнгаузом и почти одновременно Стюартом Ллойдом. Особую популярность приобрёл после работы Маккуина.

Алгоритм состоит из четырех шагов:

1) Задается число кластеров к, которое должно быть сформировано из объектов исходной выборки.

2) Случайным образом выбирается к записей исходной выборки, которые будут служить начальными центрами кластеров. Начальные точки, из которых потом «вырастает» кластер, часто называют «семенами» (от англ. seeds - семена).

3) Для каждой записи исходной выборки определяется ближайший к ней центр кластера.

4) Производится вычисление центроидов - центров тяжести кластеров. Это делается путем простого определения среднего для значений каждого признака для всех записей в кластере. Например, если в кластер вошли три записи с наборами признаков (x_1, y_1), (x_2, y_2), (x_3, y_3), то координаты его центроида будут рассчитываться следующим образом:

$$(x, y) = \left(\frac{x_1+x_2+x_3}{3}, \frac{x_1+x_2+x_3}{3}\right)$$

Затем старый центр кластера смещается в его центроид. Таким образом, центроиды становятся новыми центрами кластеров для следующей итерации алгоритма.

Шаги 3 и 4 повторяются до тех пор, пока выполнение алгоритма не будет прервано либо не будет выполнено условие в соответствии с некоторым критерием сходимости.

Остановка алгоритма производится тогда, когда границы кластеров и расположения центроидов не перестанут изменяться от итерации к итерации, т.е.

на каждой итерации в каждом кластере будет оставаться один и тот же набор записей. На практике алгоритм k-means обычно находит набор стабильных кластеров за несколько десятков итераций.

Что касается критерия сходимости, то чаще всего используется критерий суммы квадратов ошибок между центроидом кластера и всеми вошедшими в него записями, т.е.

$$E = \sum_{i=1}^{k} \sum_{p \in C_i} (p - m_i)^2$$

Где p ϵ C_i — произвольная точка данных, принадлежащая кластеру C_i, m_i - центроид данного кластера. Иными словами, алгоритм остановится тогда, когда ошибка E достигнет достаточно малого значения.

2.2. Свойства алгоритма, вычислительная сложность

Свою популярность алгоритм *k-means* приобрел благодаря следующим свойствам:

1) умеренные вычислительные затраты, которые растут линейно с увеличением числа записей исходной выборки данных. Вычислительная сложность алгоритма определяется как: $O(n) = k * n * l$

   где k - число кластеров, n - число записей и l - число итераций. Поскольку k и l заданы, то вычислительные затраты возрастают пропорционально числу записей исходного множества.

2) результаты его работы не зависят от порядка следования записей в исходной выборке, а определяются только выбором исходных точек.

Одним из основных недостатков, присущих этому алгоритму, является отсутствие четких критериев выбора числа кластеров, целевой функции их инициализации и модификации. Кроме этого, он является очень чувствительным к шумам и аномальным значениям в данных, поскольку они способны значительно повлиять на среднее значение, используемое при вычислении положений центроидов. Чтобы снизить влияние таких факторов, как шумы и аномальные значения, иногда на каждой итерации используют не среднее значение признаков, а их медиану. Данная модификация алгоритма называется k-mediods (k-медиан).

Пример работы алгоритма k-means, разбивающего объекты на 5 кластеров, приведен на рис. 1. Центры кластеров отмечены знаком «X», а каждому кластеру соответствует свой цвет.

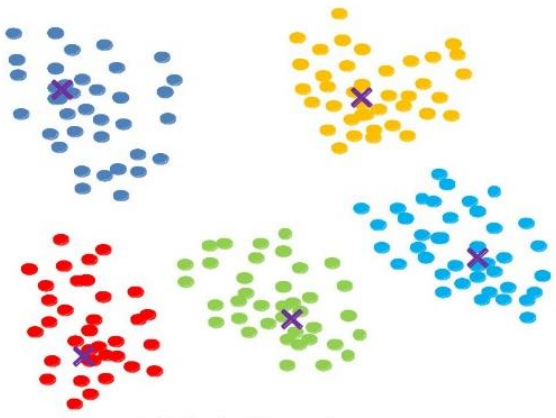

Рис. 1

2.3. Расстояние Манхэттена и Евклидово расстояние

Ключевым моментом алгоритма *k-means* является вычисление на каждой итерации расстояния между записями и центрами кластеров, что необходимо для определения, к какому из кластеров принадлежит данная запись. Правило, по которому производится вычисление расстояния в многомерном пространстве признаков, называется метрикой. Наиболее часто используются следующие метрики:

• Евклидово расстояние (метрика 1). Использует для вычисления расстояний следующее правило:

$$d_e = (x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Где x = (x_1, x_2, ..., x_m), y = (y_1, y_2, …, y_m) наборы (векторы) значений признаков двух записей. Поскольку множество точек, равноудаленных от некоторого центра при использовании евклидовой метрики будет образовывать сферу (или круг в двумерном случае), то и кластеры, полученные с использованием евклидова расстояния, также будут иметь форму, близкую к сферической.

• Расстояние Манхэттена (метрика 2). Фактически это кратчайшее расстояние между двумя точками, пройденное по линиям, параллельным осям координатной системы (рис. 2).

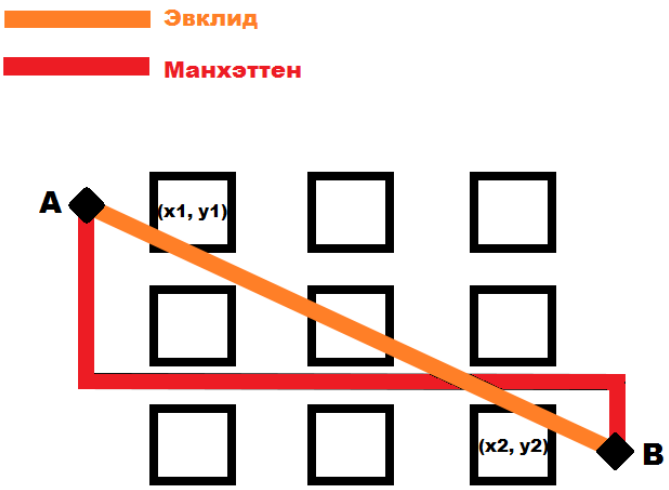

Рис. 2

Преимущество метрики 2 заключается в том, что ее использование позволяет снизить влияние аномальных значений на работу алгоритмов. Кластеры, построенные на основе расстояния Манхэттена, стремятся к кубической форме.

2.4. Пример работы алгоритма

Пусть имеется набор из 8 точек данных в двумерном пространстве, из которого требуется получить два кластера. Значения точек приведены в таблице 1.

A	B	C	D	E	F	G	H
(1; 3)	(3; 3)	(4; 3)	(5; 3)	(1; 2)	(4; 2)	(1; 1)	(2; 1)

Табл.1

Рассмотрим действия по шагам:

1. Определим число кластеров, на которое требуется разбить исходное множество $к = 2$.
2. Случайным образом выберем две точки, которые будут начальными центрами кластеров $m_l = (l; l)$ и $m_2 = (2; 1)$.
3. Проход 1. Для каждой точки определим к ней ближайший центр кластера с помощью расстояния Евклида. Таким образом, получим расстояния между центрами кластеров $m_l = (l; 1)$, $m_2 = (2; l)$ и каждой точкой исходного множества, а также в таблице 2 указано, к какому кластеру принадлежит та или иная точка.

Точка	Расстояние от m_1	Расстояние От m_2	Принадлежит к кластеру
A	2,00	2,24	1
B	2,83	2,24	2
C	3,61	2,83	9
D	4,47	3,61	2
E	1,00	1,41	1
F	3,16	2,24	2
G	0,00	1,00	1
H	1,00	0,00	2

Табл.2

Таким образом, кластер 1 содержит точки A, E, G, а кластер 2 - точки B, C, D, F, H. Как только определятся члены кластеров, может быть рассчитана сумма

квадратичных ошибок:

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} (p - m_i)^2 = 2^2 + 2{,}24^2 + 2{,}83^2 + 3{,}61^2 + 1^2 + 2{,}24^2 + 0^2 + 0^2 = 36$$

4. Проход 1. Для каждого кластера вычисляется его центроид, и центр кластера перемещается в него.

Центроид для первого кластера вычисляется как:

$$[(1 + 1 + 1) / 3, (3 + 2 + 1) / 3)] = (1;2)$$

Центроид для кластера 2 будет равен:

$$[(3 + 4 + 5 + 4 + 2) / 5, (3 + 3 + 3 + 2 + 1) / 5] = (3{,}6; 2{,}4)$$

3. Проход 2. После того, как найдены новые центры кластеров, для каждой точки снова определяется ближайший к ней центр и ее отношение к соответствующему кластеру. Для это еще раз вычисляются евклидовы расстояния между точками и центрами кластеров. Результаты вычислений приведены в таблице 3.

Точка	**Расстояние от m_1**	**Расстояние От m_2**	**Принадлежит к кластеру**
A	1,00	2,67	1
B	2,24	0,75	2
C	3,16	0,75	2
D	4,12	1,52	2
E	0,00	2,63	1
F	3,00	0,57	2
G	1,00	2,95	1
H	1,41	2,13	1

Табл.3

Относительно большое изменение m_2 привело к тому, что запись H

оказалась ближе к центру m_1, что автоматически сделало ее членом кластера 1. Все остальные записи остались в тех же кластерах, что и на предыдущем проходе алгоритма. Таким образом, кластер 1 будет A, E, G, H, а кластер 2 - B, C, D, F. Новая сумма квадратов ошибок составит:

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} (p - m_i)^2 = 1^2 + 0{,}85^2 + 0{,}72^2 + 1{,}52^2 + 0^2 + 0{,}57^2 + 1^2 + 1{,}41^2 = 7{,}83$$

что показывает уменьшение ошибки относительно начального состояния центров кластеров (которая на первом проходе составляла 36). Это говорит об улучшении качества кластеризации, т.е. более высокую «кучность» объектов относительно центра кластера.

4. Проход 2. Для каждого кластера вновь вычисляется его центроид, и центр кластера перемещается в него. Новый центроид для 1-го кластера вычисляется как:

$$[(1 + 1 + 1 + 2) / 4, (3 + 2 + 1 + 1) / 4)] = (1{,}25; 1{,}75)$$

Центроид для кластера 2 будет:

$$[(3 + 4 + 5 + 4) / 4, (3 + 3 + 2 + 4) / 4)] = (4; 2{,}75)$$

3. Проход 3. Для каждой записи вновь ищется ближайший к ней центр кластера. Полученные на данном проходе расстояния представлены в таблице 4.

Точка	**Расстояние от m_1**	**Расстояние От m_2**	**Принадлежит к кластеру**
A	1,27	3,01	1
B	2,15	1,03	2
C	3,02	0,25	9
D	3,95	1,03	2
E	0,35	3,09	1
F	2,76	0,75	2
G	0,79	3,47	1
H	1,06	2,66	1

Табл.4

Следует отметить, что записей, сменивших кластер на данном проходе алгоритма, не было. Новая сумма квадратов ошибок составит:

$$E = \sum_{i=1}^{k} \sum_{p \in c_i} (p - m_i)^2 = 1{,}27^2 + 1{,}03^2 + 0{,}25^2 + 1{,}03^2 + 0{,}35^2 + 0{,}75^2 + 0{,}79^2 + 1{,}06^2 = 6{,}25$$

Таким образом, изменение суммы квадратов ошибок является незначительным по сравнению с предыдущим проходом.

4. Проход 3. Для каждого кластера вновь вычисляется его центроид, и центр кластера перемещается в него. Но поскольку на данном проходе ни одна запись не изменила своего членства в кластерах, то положение центроида не поменялось, и алгоритм завершает свою работу.

Рассчитаем вычислительную сложность по формуле:

$$O(n) = к * n * l$$

Где *k* - число кластеров, *n* - число записей, *l* - число проходов:

$$O(n) = 2 * 8 * 3$$

$$O\ (n) = 48$$

3. Заключение

В результате выполнения работы были рассмотрены алгоритмы кластеризации, разработан и исследован алгоритм кластеризации k-means.

Исходя из особенностей алгоритмов кластеризации они приносят большую пользу как инструмент исследования данных. Можно построить для наших данных иерархическую структуру из нескольких уровней кластеров или сформировать заранее заданное число кластеров. Теоретически, для кластеризации пригоден любой набор данных, состоящий из объектов, которые можно определить в терминах значений атрибутов. Но требуется внимательно подходить к выбору меры расстояний между объектами и соответствующего алгоритма.

Осуществлено моделирование разработанного алгоритма на примере *k-means*. Для разбиения множества из восьми точек на 2 кластера неиерархическому, четкому алгоритму k-means понадобилось 3 прохода, что доказывает его довольно высокую эффективность для небольших множеств. В ходе рассмотрения данного алгоритма выявлены следующие достоинства:

- простота использования;
- быстрота использования;
- понятность и прозрачность алгоритма;

Недостатки алгоритма к-средних:

- алгоритм слишком чувствителен к выбросам, которые могут искажать среднее. Возможным решением этой проблемы является использование модификации алгоритма - алгоритм к-медианы;
- алгоритм может медленно работать на больших базах данных. Возможным решением данной проблемы является использование выборки данных

Список используемой литературы

1. Хараламбос Марманис Х., Бабенко Д. Алгоритмы интеллектуального Интернета. - М.: Символ-Плюс, 2011, - 480 с.
2. *Jain A, Murty M, Flynn P. Data Clustering: A Review. И ACM Computing Surveys. 1999. Vol. 31, no. 3.*
3. Интернет-ресурс сообщества /Т'-специалистов ''Хабрахабр". http://habrahabr.ru
4. Свободная энциклопедия ''Википедия". http://ru.wikipedia.org
5. Интернет-ресурс *Microsoft Developer Network* (*MSDN*). http://msdn.microsoft.com
6. Интернет-ресурс Laboratory of Mathematical Logic at PDMI

   http://logic.pdmi.ras.ru

Исходный код алгоритма *k-means*

```
#define MAX_STR_LEN 512
#define MAX_LIN_LEN 30
#define BIGNUM 1.0e12
#define PRTINC 10

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <math.h>
#include <ctype.h>


int main(int argc, char *argv[])
{
  FILE *ipFile;
  FILE *centFile;
  FILE *opFile;
  int i;
  int j;
  int k;
  float **x;
  float **cent;
  float *mean;
  float **dist;
  int *bestCent;
  char **label;
  char **cLabel;
  int it;
  int numRows;
  int numCols;
  int numCols2;
  int numCent;
  float *distortion;
  int NUMIT;

  /* Check correct number of input parameters */
  if ((argc!=5)&&(argc!=4))
    {
      printf("format: k-means ipFile centroids opFile <numIt>\n");
      exit(1);
    }
  if (argc==4) NUMIT=10;

  /* allocate space for distortion scores */
  distortion=(float *)calloc(NUMIT, sizeof(float));

  printf("open input file\n");
  /* set pointer to input data file */
  argv++;
  if ((ipFile=fopen(*argv,"r"))==NULL)
    {
      printf("Error: can't open input file %s\n",*argv);
      exit(1);
    }

  printf("open input file\n");
  /* set pointer to centroid data file */
  argv++;
```

```
if ((centFile=fopen(*argv,"r"))==NULL)
  {
    printf("Error: can't open input file %s\n",*argv);
    exit(1);
  }

printf("open output file\n");
/* set pointer to outnput data file */
argv++;
if ((opFile=fopen(*argv,"w"))==NULL)
  {
    printf("Error: can't open output file %s\n",*argv);
    exit(1);
  }

/* set pointer to number of iterations */
argv++;
NUMIT=atoi(*argv);
printf("Number of iterations = %d\n",NUMIT);

/* read number of data points */
if(fscanf(ipFile,"# num rows=%d num columns=%d\n",&numRows,&numCols)!=2)
  {
    printf("Format error in first line\n");
    exit(1);
  }

printf("numRows=%d numCols=%d\n",numRows,numCols);

/* allocate memory */
x=(float **)calloc(numRows,sizeof(float *));
dist=(float **)calloc(numRows,sizeof(float *));
label=(char **)calloc(numRows,sizeof(char *));
mean=(float *)calloc(numCols,sizeof(float));
for (j=0; j<numRows; j++)
  {
    x[j]=(float *)calloc(numCols,sizeof(float));
    label[j]=(char *)calloc(MAX_STR_LEN,sizeof(char));
  }
i=numRows;

/* read in labels and data points */
for (j=0; j<numRows; j++)
  {
    int k;
    /* first read in label */
    fscanf(ipFile,"%s ",label[j]);
    for (k=0; k<numCols; k++)
      {
        fscanf(ipFile,"%f ",&(x[j][k]));
        mean[k]+=x[j][k];
      }
    fscanf(ipFile,"\n");
  }

printf("number of data points = %d\n",numRows);
for (k=0; k<numCols; k++) mean[k]/=numRows;

/* get number of centroids */
if(fscanf(centFile,"# num rows=%d num columns=%d\n",&numCent,&numCols2)!=2)
  {
```

```
    printf("Format error in first line\n");
    exit(1);
  }
if (numCols!=numCols2)
  {
    printf("Incompatible matrices - different numbers of columns\n");
    exit(1);
  }

printf("numCentroid=%d, numCols=%d\n",numCent,numCols2);

/* allocate memory */
cent=(float **)calloc(numCent,sizeof(float *));
cLabel=(char **)calloc(numCent,sizeof(char *));
bestCent=(int *)calloc(numRows,sizeof(int));
for (j=0; j<numRows; j++) dist[j]=(float *)calloc(numCent,sizeof(float));
for (j=0; j<numCent; j++)
  {
    cent[j]=(float *)calloc(numCols,sizeof(float));
    cLabel[j]=(char *)calloc(MAX_STR_LEN,sizeof(char));
  }
i=numRows;

/* read in centroid labels and data points */
for (j=0; j<numCent; j++)
  {
    int k;
    /* first read in label */
    fscanf(centFile,"%s ",cLabel[j]);
    /* then read in data values */
    for (k=0; k<numCols; k++)
      {
        fscanf(centFile,"%f ",&(cent[j][k]));
      }
    fscanf(ipFile,"\n");
  }

/* write centroids to file */
fprintf(opFile,"original\n");
for (j=0; j<numCent; j++)
  {
    int k;
    for (k=0; k<numCols; k++) fprintf(opFile,"%f ",x[j][k]);
    fprintf(opFile,"\n");
  }

for (it=0; it<NUMIT; it++)
  {
    /* calculate distance matrix and minimum */
    float rMin;
    int count;

    distortion[it]=0;
    count=0;

    for (j=0; j<numRows; j++)
      {
        rMin=BIGNUM;
        for (k=0; k < numCent; k++)
          {
            int e;
```

```
            dist[j][k]=0;
            for (e=0; e<numCols; e++)
              {
                dist[j][k]+=(x[j][e]-cent[k][e])*(x[j][e]-cent[k][e]);
                                }
            /* printf("x[j]=(%f,%f), cent[k]=(%f,%f),
dist[%d][%d]=%f\n",x[j][0],x[j][1],cent[k][0],cent[k][1],j,k,dist[j][k]); */
            if (dist[j][k] < rMin)
              {
                bestCent[j]=k;
                rMin=dist[j][k];
              }
          }
        distortion[it]+=rMin;
        count++;
      }

    /* distortion[it]=distortion[it]/count; */

    /* reestimate centroids */
    for (k=0; k<numCent; k++)
      {
        float *cent_r;
        int tot;

        cent_r=(float *)calloc(numCols,sizeof(float));
        tot=0;
        for (j=0; j<numRows; j++)
          {
            if (bestCent[j]==k)
              {
                int e;
                for (e=0; e<numCols; e++) cent_r[e]+=x[j][e];
                tot++;
              }
          }

        if (tot > 0)
          {
            int e;
            for (e=0; e<numCols; e++) cent[k][e]=cent_r[e]/tot;
          }
        else
          {
            int e;
            for (e=0; e<numCols; e++) cent[k][e]=mean[e];
          }
      }

    /* write centroids to file */
    fprintf(opFile,"\n\nit=%d, distortion=%f\n",it,distortion[it]);
    for (k=0; k<numCent; k++)
      {
        int e;
        for (e=0; e<numCols; e++) fprintf(opFile,"%f ",cent[k][e]);
        fprintf(opFile,"\n");
      }
    for (k=0; k<numCent; k++)
      {
        fprintf(opFile,"Cluster %d\n",k);
        for (j=0; j<numRows; j++)
```

```
            {
              if (bestCent[j]==k) fprintf(opFile,"%s\n",label[j]);
            }
        }
    }

  /* write clusters to screen */
  printf("Results\n=======\n\n");
  for (it=0; it<NUMIT; it++) printf("distortion[%d]=\t%f\n",it,distortion[it]);

  for (k=0; k<numCent; k++)
    {
      printf("\nCluster %d\n=========\n",k);
      for (j=0; j<numRows; j++)
        {
          if (bestCent[j]==k) printf("%s\n",label[j]);
        }
    }

  return(0);
}
```