Homework 2
Computer Science
Fall 2015
B565

Suhas Jagadish
Sunday, October 18, 9:00 p.m.

October 18, 2015

All the work herein is solely mine

1 k-means Algorithm in Theory

ALGORITHM `kmeans`
INPUT (data Δ, distance $d : \Delta^2 \to \mathbb{R}_{\geq 0}$, centoid number k, threshold τ)
OUTPUT (Set of centoids $c_1, c_2, \ldots, c_k$)
Assume centroid is structure $c = (v \in DOM(\Delta), B \subseteq \Delta)$
$c.v$ is the centroid value and $c.B$ is the set of nearest points.
τ is a percentage change from previous centroids
For example, $\{c_1, c_2, \ldots, c_k\}$ is previous and $\{d_1, d_2, \ldots, d_k\}$ is current
Total difference is $\Sigma_i \Sigma_j d(c_i, d_j)$
$Dom(\Delta)$ denotes domain of object.
$i = 0$ ▷ Initialize iterate where superscript is iteration
for $j = 1, k$ **do** ▷ Initialize Centroids
    $c_j^i.v \leftarrow random(Dom(\Delta))$
    $c_j^i.B \leftarrow \emptyset$
end for
$f_i = \Sigma_{j=1}^k \Sigma_{\ell=1}^k d(c_j^i.v, random(Dom(\Delta))$ ▷ Bootstrap difference between past centroids and current
repeat
    $i \leftarrow i + 1$
    for $\delta \in \Delta$ **do**
        $c_j^i.B \leftarrow c.B \cup \{\delta\}$, where $\min_{c_j^i}\{d(\delta, c_j^i.v)\}$
        ▷ Associate a data point δ with the nearest centroid $c_j^i.v$
    end for
    for $j = 1, k$ **do**
        $c_j^i.v \leftarrow ave(c_j^i.B)$ ▷ Update centroid to be *best* representative of nearest data
        $c_j^i.B \leftarrow \emptyset$ ▷ ave is easiest representative
    end for
    $f_i \leftarrow \Sigma_{j=1}^k \Sigma_{\ell=1}^k d(c_j^i.v, c_\ell^{i-1})$ ▷ Find the difference between previous centroids and current centroids
until $(|f_i - f_{i-1}| < \tau(f_{i-1}))$
return $(c_1^i, c_2^i, \ldots, c_3^i)$ ▷ If there's less than τ change, then we're finished.

1.1 Questions

1. Does the algorithm always converge? Given your answer, what extra formal parameter is needed to the function. How do you decide it's actual value?
   Solution: Not necessarily. In the above algorithm, there may be a possibility where $|f_i - f_{i-1}| < \tau(f_{i-1})$ is never satisfied and the algorithm may not converge.

   In order to make the algorithm converge, we use an additional parameter, "an iterator".

   There is no definite way to decide the actual value of the number of iterations to be performed.
   One way to decide is, assume the number of iterations to be a specific value, say 50. We run k-means 50 times, calculate the threshold for each iteration, pick the average threshold. Now we change the number of iterations to, say 100. We run k-means again 100 times and get the average threshold. We repeat this process for a fair number of times and record all the average threshold values.
   We then check to see which threshold value is near to the actual expected threshold value(one we set initially in our algorithm) and pick the corresponding number of iterations. This is one way of deciding the number of iterations to be performed on k-means for convergence.

2. What is the reason initialization of k-means is problematic?
   Solution: Initialization of k-means may result in the following problems-
   - We may end up with different outcomes(set of centroids) each time a random initialization is made.
   - Convergence is not guaranteed - random initialization may result in outliers which typically increase the sum of squared error(SSE).
   - Randomly chosen initial set of centroids can be too close to one another.
   - Bad initialization can lead to poor convergence speed and hence may overall result in bad clustering.

3. What is the run-time of this algorithm (include your new parameter from Question 1.
   Solution: Including "iterator" as a parameter, the run-time of this algorithm would be,

   $$O(I * K * m * n)$$

   where I = number of iterations required for convergence, K = number of centroids chosen, m = number of data points(size of the data) and n = number of attributes defined.

4. In line 12, $f_i = 2d((0,0), c_1^0) + 2d((0,0), c_2^0)$. Why are the distances multiplied by 2?
   Question not valid

5. In line 23, we use *ave* to indicate average; however, the centroid is more correctly the best representative of the data that nearest. Give two more ways (functions) that yield a best representative.
   Solution: While clustering, as a thumb rule, we need to keep in mind to maintain minimum interblock distance and maximum intrablock distance.

   So the first way is to find a datapoint that is nearest to the centroid. We calculate the distance from the centroid to each data point in a cluster. We then pick the datapoint which is at a minimum distance from the centroid and then choose it as the new centroid. This maintains the minimum interblock distance in a cluster.

   Second way is to measure the distances between each pair of data points in the 2 clusters and pick the data points which are farthest from one another.
   We calculate the distance between each data point in one cluster to every other data point in another cluster. We then pick 2 data points whose distance is maximum. This maintains the maximum interblock distance between 2 clusters.

6. Modify the algorithm to identify when two centroids are *too* close to one another. There will likely be more than one extra parameter needed to the algorithm. Discuss what your modification does.
Solution: We introduce an additional parameter, threshold or minimum distance.
We calculate the distance between the 2 centroids and if the distance is less than the minimum distance, we either merge the 2 centroids or re-initialize them.
We need this check to be performed initially when the random centroids are chosen(to make sure they are not chosen very close to each other) and when the new set of centroids are calculated.
Initially when the 2 randomly chosen centroids are too close, then they are re-initialized until the distance is greater than minimum distance(line 9 to line 16).
Later when the new set of centroids formed are too close, we merge both the centroids, assign the related to datapoints to merged centroid and decrement the number of centroids(line 28 to 36).
The modification in the algorithm is below-

1: **ALGORITHM** `kmeans`
2: **INPUT** (data Δ, distance $d : \Delta^2 \to \mathbb{R}_{\geq 0}$, centoid number k, threshold τ, minimum distance)
3: **OUTPUT** (Set of centoids $c_1, c_2, \ldots, c_k$)
4: $i = 0$ ▷ Initialize iterate where superscript is iteration
5: **for** $j = 1, k$ **do** ▷ Initialize Centroids
6: $c_j^i.v \leftarrow random(Dom(\Delta))$
7: $c_j^i.B \leftarrow \emptyset$
8: **end for**
9: **for** $j = 1, k-1$ **do**
10: **for** $l = i+1, k$ **do**
11: **while** $d(c_j^i, c_l^i) < minimum\ distance$ **do**
12: $c_l^i.v \leftarrow random(Dom(\Delta))$
13: $c_j^i.B \leftarrow \emptyset$
14: **end while**
15: **end for**
16: **end for**
17: $f_i = \Sigma_{j=1}^{k} \Sigma_{\ell=1}^{k} d(c_j^i.v, random(Dom(\Delta))$ ▷ Bootstrap difference between past centroids and current
18: **repeat**
19: $i \leftarrow i+1$
20: **for** $\delta \in \Delta$ **do**
21: $c_j^i.B \leftarrow c.B \cup \{\delta\}$, where $\min_{c_j^i}\{d(\delta, c_j^i.v)\}$
22: ▷ Associate a data point δ with the nearest centroid $c_j^i.v$
23: **end for**
24: **for** $j = 1, k$ **do**
25: $c_j^i.v \leftarrow ave(c_j^i.B)$ ▷ Update centroid to be *best* representative of nearest data
26: $c_j^i.B \leftarrow \emptyset$ ▷ ave is easiest representative
27: **end for**
28: **for** $j = 1, k-1$ **do**
29: **for** $l = i+1, k$ **do**
30: **if** $d(c_j^i, c_l^i) < minimum\ distance$ **then**
31: $c_j^i.v \leftarrow (c_j^i + c_l^i)/2$
32: $c_j^i.B \leftarrow c_j^i.B \cup c_l^i.B$
33: $k \leftarrow k-1$ ▷ remove c_l^i from the set of centroids
34: **end if**
35: **end for**
36: **end for**
37: $f_i \leftarrow \Sigma_{j=1}^{k} \Sigma_{\ell=1}^{k} d(c_j^i.v, c_\ell^{i-1})$ ▷ Find the difference between previous centroids and current centroids

38: **until** $(|f_i - f_{i-1}| < \tau(f_{i-1}))$
39: **return** $(c_1^i, c_2^i, \ldots, c_3^i)$ ▷ If there's less than τ change, then we're finished.

7. Let $x = \{a, b, c, d\}, y = \{a, b, e\}, z = \{b, f\}, \mathcal{U} = \{a, b, c, d, e, f\}$. Compute the distances using

$$d(x, y) = \begin{cases} 0, & x = y \\ 1, & x \neq y \end{cases} \quad (1)$$

The signature of the distance function is: $d : \mathsf{Set}^2 \to \mathbb{R}_{\geq 0}$.

(a) $\neg x = \mathcal{U} - \{a, b, c, d\} = \{e, f\}$
(b) $\neg \mathcal{U} = \emptyset$
(c) $d(x, y) = 1$
(d) $d(x \cap y, \{a, b\}) = 0$
(e) $d(x, x \cup y) = d(\{a, b, c, d\}, \{a, b, c, d, e\}) = \mathbf{1}$
(f) $d(\neg(x \cap y), \neg x \cup \neg y) = d(\{c, d, e, f\}, \{c, d, e, f\}) = \mathbf{0}$

$$\begin{aligned} J(x, y) &= |x \cap y|/|x \cup y| \\ d(x, y) &= 1 - J(x, y) \end{aligned}$$

The signature of the distance function is: $d : \mathsf{Set}^2 \to \mathbb{R}_{\geq 0}$.

(a) $d(x, y) = 1 - |\{a, b\}|/|\{a, b, c, d, e\}| = 1 - 2/5 = \mathbf{3/5}$, assuming $|X|$ = number of elements in X.
(b) $d(x \cap y, \{a, b\}) = 1 - |\{a, b\} \cap \{a, b\}|/|\{a, b\} \cup \{a, b\}| = 1 - 2/2 = 1 - 1 = \mathbf{0}$
(c) $d(x, x \cup y) = 1 - |\{a, b, c, d\} \cap \{a, b, c, d, e\}|/|\{a, b, c, d\} \cup \{a, b, c, d, e\}| = 1 - 4/5 = \mathbf{1/5}$
(d) $d(\neg(x \cap y), \neg x \cup \neg y) = 1 - |\{c, d, e, f\} \cap \{c, d, e, f\}|/|\{c, d, e, f\} \cup \{c, d, e, f\}| = 1 - 4/4 = 1 - 1 = \mathbf{0}$

8. Assume bit vectors (as strings of 0's and 1's) over the set $\{a, b, c, d, e, f\}$. Then $x = \{a, b, c, d\}$ is $\mathbf{x} = 111100$. Similarly, $\mathbf{y} = 110010, \mathbf{z} = 010001$. Individually, $\mathbf{x}[0] = 1, \mathbf{x}[1] = 1$, *etc.* To remind the student, $\perp$ means the program stopped because of a bad computation. Let the distance function be the Hamming distance between the vector:

$$c(x, y) = \begin{cases} 0, & x = y \\ 1, & otherwise \end{cases} \quad \text{for individual characters} \quad (2)$$

$$d(\mathbf{x}, \mathbf{y}) = \Sigma_{i=0}^{n} c(\mathbf{x}[i], \mathbf{y}[i]) \quad n = ||\mathbf{x}||, \text{the length of the string.} \quad (3)$$

The signature of the distance function is: $d : \mathsf{String}^2 \to \mathbb{R}_{\geq 0}$. Assume we have some string functions and a constant:

$$\begin{aligned} \text{␣} &= \text{space} && (4) \\ concat(\texttt{b}, \texttt{at}) &= \texttt{bat} && (5) \\ concat(\texttt{bat}, \epsilon) &= \texttt{bat} && (6) \\ contat(\epsilon, \texttt{bat}) &= \texttt{bat} && (7) \\ upper(\texttt{a}) &= \texttt{A} && (8) \\ space(\texttt{b␣a␣t}) &= \texttt{bat} && (9) \end{aligned}$$

(a) $d(\mathtt{x}, \mathtt{y}) = d(111100, 110010) = 0 + 0 + 1 + 1 + 1 + 0 = \mathbf{3}$
(b) $d(concat(\mathtt{x}, \mathtt{x}), concat(\mathtt{x}, \mathtt{y})) = d(\mathtt{xx}, \mathtt{xy}) = d(111100111100, 111100110010) = 0 + 0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 + 1 + 1 + 0 = \mathbf{3}$

(c) Discuss the two problems above.
Solution: The first half of the bit vectors to compare is `x` and `x` whose distance is 0. So effectively the distance of the bit vectors is same as the previous problem, $d(\mathtt{x}, \mathtt{y})$. The distance value may be same but this does not mean both the bit vectors are same.

(d) $d(\mathtt{N\_orth}, \mathtt{nort\_h})$ = We calculate c(x,y) for each pair,
$c(N, n) = 1, c(\_, o) = 1, c(o, r) = 1, c(r, t) = 1, c(t, \_) = 1, c(h, h) = 0$
$1 + 1 + 1 + 1 + 1 + 0 = \mathbf{5}$

(e) $d(upper(space(\mathtt{N\_orth})), upper(space(\mathtt{nort\_h}))) = d(upper(North), upper(north)) = d(NORTH, NORTH) = 0 + 0 + 0 + 0 + 0 = \mathbf{0}$

(f) $d(\mathtt{north}, \mathtt{south}) = 1 + 0 + 1 + 0 + 0 = \mathbf{2}$

(g) Because strings are seldom fixed to any length, finding distance is even more difficult. strings of unequal length. The table below shows *some* strings indicating the direction prefixed or suffixed to addresses:

NORTH
North
N
N.
north
nor
n.
_n

So the original distance would simply not function, *e.g.,*

$$d(\mathtt{N.}, \mathtt{North}) \quad = \quad \perp \tag{10}$$

How would you rewrite the distance function to effectively deal with this problem?
Solution: We have several ways to deal with this problem, just to make sure that the program doesn't stop. However our aim is to identify some sort of similarity between the 2 strings.
One way is to calculate the minimum of the lengths of the strings, then compare each element of the 2 strings up to the minimum length,

$$d(\mathbf{x}, \mathbf{y}) \quad = \quad \Sigma_{i=0}^{min(||\mathbf{x}||, ||\mathbf{y}||)} c(\mathbf{x}[i], \mathbf{y}[i]) \quad where ||\mathbf{x}|| = \text{the length of the string.} \tag{11}$$

So the distance $d(\mathtt{N.}, \mathtt{North})$ would evaluate to $d(\mathtt{N.}, \mathtt{No}) = 0 + 1 = \mathbf{1}$.
In terms of similarity, I'm defining my function such that "." is not entertained while indicating directions.
There can be another distance function which just checks the first letter of each string and if they are equal return 0. This would also not break the code. Problem with this approach is, suppose we have a distance $d(\mathtt{Not}, \mathtt{North})$. In this case, the value returned is 0 which says strings are equal but ideally "Not" and "North" are not the same.

9. Assume your data is a set and string pair, $\delta = (\mathsf{Set}\ x, \mathsf{String}\ y)$. Create a metric over this pair. In otherwords,

$$d((\mathsf{Set}\ x_1, \mathsf{String}\ y_1), (\mathsf{Set}\ x_2, \mathsf{String}\ y_2)) \quad =$$

Demonstrate that it is indeed a metric.

Solution: Let us assume the 'Set' to be a finite list of alphabets and 'String' to be a string from that Set. For example,
Let Set $x_1 = \{a, b, c, d, h, s, u, z\}$ and Set $x_2 = \{i, j, k, l, m, n, o, t\}$

Let String $y_1 = \{"suhas"\}$ and String $y_2 = \{"tim"\}$

In order to compare a set and string pair, we need to first find the distance between 2 sets and distance between 2 strings. Then we build the actual distance function for the set-string pair using the above individual distances.

$$d_1(x1, x2) = \begin{cases} 0, & \forall x1_i = x2_i, \quad where\ i = 1\ to\ min(||x1||, ||x2||) \\ 1, & otherwise \end{cases} \tag{12}$$

$$d_2(y1, y2) = \begin{cases} 0, & \forall y1_i = y2_i, \quad where\ i = 1\ to\ min(||y1||, ||y2||) \\ 1, & otherwise \end{cases} \tag{13}$$

$$\tag{14}$$

So my actual distance function would be,

$$d(\mathbf{x}, \mathbf{y}) = d_1(x1, x2) + d_2(y1, y2), \quad where\ \mathbf{x} = (x1, y1)\ and\ \mathbf{y} = (x2, y2) \tag{15}$$

Basically my distance function between 2 set-string pairs is simply the addition of distances between 2 sets and distances between 2 strings. Now let us prove that the actual distance function is a metric.

Reflexive property:
$\forall\, \mathbf{x}, d(\mathbf{x}, \mathbf{x}) = 0$
$d(\mathbf{x}, \mathbf{x}) = d(x1, x2) + d(y1, y2) = 0 + 0 = 0$

$\forall\, \mathbf{x}, \mathbf{y}, d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
Assume d(x1,x2) = 0 and d(y1,y2) = 0. Then $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) = 0$
Assume d(x1,x2) = 0 and d(y1,y2) = 1. Then $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) = 1$
Assume d(x1,x2) = 1 and d(y1,y2) = 0. Then $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) = 1$
Assume d(x1,x2) = 1 and d(y1,y2) = 1. Then $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) = 2$

$\forall\, \mathbf{x}, \mathbf{y}, \mathbf{z}, d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$
Assume d(x1,x2) = 0, d(y1,y2) = 0, d(z1,z2) = 0. Then $d(\mathbf{x}, \mathbf{y}) = 0, d(\mathbf{y}, \mathbf{z}) = 0, d(\mathbf{x}, \mathbf{z}) = 0$. So $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), since\ 0 + 0 >= 0$.
Assume d(x1,x2) = 0, d(y1,y2) = 1, d(z1,z2) = 0. Then $d(\mathbf{x}, \mathbf{y}) = 1, d(\mathbf{y}, \mathbf{z}) = 1, d(\mathbf{x}, \mathbf{z}) = 0$. So $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), since\ 1 + 1 >= 0$.
Assume d(x1,x2) = 0, d(y1,y2) = 0, d(z1,z2) = 1. Then $d(\mathbf{x}, \mathbf{y}) = 0, d(\mathbf{y}, \mathbf{z}) = 1, d(\mathbf{x}, \mathbf{z}) = 1$. So $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), since\ 0 + 1 >= 1$.
Assume d(x1,x2) = 0, d(y1,y2) = 1, d(z1,z2) = 1. Then $d(\mathbf{x}, \mathbf{y}) = 1, d(\mathbf{y}, \mathbf{z}) = 2, d(\mathbf{x}, \mathbf{z}) = 1$. So $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), since\ 1 + 2 >= 1$.

Assume d(x1,x2) = 1, d(y1,y2) = 0, d(z1,z2) = 0. Then $d(\mathbf{x}, \mathbf{y}) = 1, d(\mathbf{y}, \mathbf{z}) = 0, d(\mathbf{x}, \mathbf{z}) = 1$. So $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), since\ 1 + 0 >= 1$.
Assume d(x1,x2) = 1, d(y1,y2) = 1, d(z1,z2) = 0. Then $d(\mathbf{x}, \mathbf{y}) = 2, d(\mathbf{y}, \mathbf{z}) = 1, d(\mathbf{x}, \mathbf{z}) = 1$. So $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), since\ 2 + 1 >= 1$.
Assume d(x1,x2) = 1, d(y1,y2) = 0, d(z1,z2) = 1. Then $d(\mathbf{x}, \mathbf{y}) = 1, d(\mathbf{y}, \mathbf{z}) = 1, d(\mathbf{x}, \mathbf{z}) = 2$. So $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), since\ 1 + 1 >= 2$.
Assume d(x1,x2) = 1, d(y1,y2) = 1, d(z1,z2) = 1. Then $d(\mathbf{x}, \mathbf{y}) = 2, d(\mathbf{y}, \mathbf{z}) = 2, d(\mathbf{x}, \mathbf{z}) = 2$. So $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}), since\ 2 + 2 >= 2$.

Hence the distance function $d(\mathbf{x}, \mathbf{y})$ is proved to be a metric considering the assumptions made above.

2 Application of k-means to medical data

This problem examines Wolberg's breast cancer data[**?**]. The data is found at
`http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/`

data	`breast-cancer-wisconsin.data`
description	`breast-cancer-wisconsin.names`

1. Suppose you're working to help a clinic serve a community that has limited resources to identify then treat breast cancer. The cost of a biopsy is from \$1000 to \$5000. The cost of a masectomy is \$15,000 to \$55,000 (these are representative costs in 2015).

   (a) What was the total cost of the biopsies?
   Solution: There are 8 duplicate rows in the provided data set out of which 5 datasets have label as 2 and 3 datasets with label 4. So we first remove the duplicate rows in R and import our dataset. Our dataset now contains 691 rows of data(453 rows with label as 2 and 238 rows with label as 4).
   Irrespective of the label value, all the patients would have taken biopsy test. Hence the total cost of biopsies would range from \$6,91,000 to \$34,55,000. Average total cost of biopsies would be \$20,73,000.

   (b) What would have been the likely total cost of masectomies?
   Solution: We have 238 rows of data in our dataset with label as 4. Label 4 denotes that masectomy has to performed on these 238 rows.
   Hence the total cost of masectomies would range from \$35,70,000 to \$130,90,000. Average total cost of masectomies would be \$83,30,000.

2. Ignoring the `Sample code number` (SCN), how many attributes does Δ have?
   Solution: If we consider label as an attribute, then ignoring Sample code number(SCN), Δ has 10 attributes. If we do not consider label as an attribute, then ignoring Sample code number(SCN), Δ has 9 attributes.

3. How many missing values exist (total)? There are **16 missing values** in total.

   How many patients have missing values? For each row of data, we have Sample code number(SCN) denoting the key value for each patient. From the data, we know that **16 patients have missing values**.

   Give the SCNs for that have missing data. **SCNs for missing data are** 1057013, 1096800, 1183246, 1184840, 1193683, 1197510, 1241232, 169356, 432809, 563649, 606140, 61634, 704168, 733639, 1238464 and 1057067.

   Of these data, would you have recommended re-examination for the women? What would be the cost be? - **Yes**. We do not know the actual impact of bare nuclei on determining the nature of benign/malignant cell. With this uncertainty, we do not want to predict something wrong. Hence I would recommend re-examination for those missing data.
   Cost for this would range from \$16,000 to \$80,000 with an average of \$48,000.

   What is the error rate (in otherwords, given x as the number of patients, what is $f(x) = y$ where y is the number of mistakes). - In our dataset, we have multiple rows of data which contains same SCN i.e., multiple rows for same patient. So we need to count unique number of patients(SCNs) in the data set, which is 645. Out of 645 patients, 16 of them have mistakes. So the error rate is (16/645)*100 = **2.48%**.

   Remove the tuples that have missing data. Let Δ^* be a cleaned Δ: the tuples with the missing

values are removed. R offers several ways to remove unknown data, though you are free to write your own code. - I used the R function na.omit(read.csv("meddata_na.csv")) to remove tuples that have missing data.

Let $\Delta^m = \Delta - \Delta^*$. For each $\delta \in \Delta^m$, replace the unknown data using one of the techniques we discussed in class; alternatively, you may employ your own approach. No matter how you decide to replace the unknowns, explain fully. The final data should be presented as $(\texttt{SCN}, A_i, data)$ where SCN is the tuple key, A_i is the attribute, and *data* is the new data.

Solution: I thought of 2 ways to replace the unknown data. First, I would replace the missing values with the most occurring value in that column. Most occurring value in "Bare nuclei" column is 1, hence I replace all the missing values in "Bare nuclei" column with 1.
Second, I try to correlate all the columns in my cleaned dataset Δ^* using R function $cor(data)$. Using the correlation table, I found that column "Bare nuclei" is closest to column "Uniformity of cell shape". The correlation table is attached as "correlation.xls".
Now I consider my actual dataset Δ containing the missing values, filter the missing values and replace all of them with the corresponding values from column "Uniformity of cell shape".
I calculated mean and variance of the column "Bare nuclei" after replacing the missing values in both the above cases. The results are shown below -
Replacing the most occurring value, Variance = 13.118, mean = 3.486
Replacing the corresponding values of column "Uniformity of cell shape", Variance = 13.086, mean = 3.529

This suggests that both the approaches are somewhat similar and hence I choose to replace the missing values with corresponding values of column "Uniformity of cell shape". Final table in $(\texttt{SCN}, A_i, data)$ format would be -

SCN	A_7	data
1057013	Bare Nuclei	5
1096800	Bare Nuclei	6
1183246	Bare Nuclei	1
1184840	Bare Nuclei	3
1193683	Bare Nuclei	2
1197510	Bare Nuclei	1
1241232	Bare Nuclei	4
169356	Bare Nuclei	1
432809	Bare Nuclei	3
563649	Bare Nuclei	8
606140	Bare Nuclei	1
61634	Bare Nuclei	3
704168	Bare Nuclei	5
733639	Bare Nuclei	1
1238464	Bare Nuclei	1
1057067	Bare Nuclei	1

(a) Is the amount of missing data significant?
Solution: The error rate is 2.48% and I believe that the amount of missing data is significant looking at the error rate.

(b) Assess the significance of either keeping or removing the tuples with unknown data. You should consider both the morbidity and cost.
Solution: If we remove missing or unknown data, we have to send it for re-examination which would result in additional cost of $16,000 to $80,000(assuming the cost of biopsy test).
However the presence of bare nuclei(attribute of the missing data) may be helpful in interpreting

the benign or malignant nature of the cell. Or may be in identifying the benign or malignant type.
If we keep the tuples with missing data by substituting values as discussed above, we might misinterpret the results. In the medical field, we need to have accurate results especially with sensitive diseases such as breast cancer.
Hence considering the impact of unknown data, I would remove such tuples and send it for re-examination, choosing morbidity over cost.

4. Assume the attribute `Clump Thickness` is A_1, `Uniformity of Cell Size` is A_2 and so on. Attribute A_{10} has only two domain values and is the classifier. For Δ^* and the attributes $A_i, 1 \leq i \leq 9$

   (a) which A_i has the greatest variance? You will write an `R` function that takes a list of numbers and returns the variance.
   Solution: The R function to calculate the variance of an attribute is attached in the "Variance.R" file.
   First, I clean the data to remove all the duplicates and missing values. Then I select only 9 attributes starting from `Clump Thickness` to `Mitosis`. Then I call my variance function which returns the variance of that attribute. The values are -
   Variance of `Clump Thickness` is 7.957248
   Variance of `Uniformity of cell size` is 9.333056
   Variance of `Uniformity of cell shape` is 8.859862
   Variance of `Marginal Adhesion` is 8.2709
   Variance of `Single Epithelial cell size` is 4.877459
   Variance of `Bare nuclei` is 13.2341
   Variance of `Bland Chromatin` is 6.021594
   Variance of `Normal nucleoli` is 9.397947
   Variance of `Mitosis` is 3.031102

   As we can see from the above list of values, attribute "Bare nuclei" has the greatest variance.

   (b) which A_i has the lowest entropy? You may use the `R` package `entropy` by Hausser and Strimmer.
   Solution First we need to install `entropy` package into R. Then import the package into our code. The R code to calculate entropies of each attribute is attached in "entropy.R" file. The output of the code is,
   Entropy of `Clump Thickness` is 3.048984
   Entropy of `Uniformity of cell size` is 2.349703
   Entropy of `Uniformity of cell shape` is 2.494103
   Entropy of `Marginal Adhesion` is 2.221525
   Entropy of `Single Epithelial cell size` is 2.283391
   Entropy of `Bare nuclei` is 1.995922
   Entropy of `Bland Chromatin` is 2.773736
   Entropy of `Normal nucleoli` is 2.058127
   Entropy of `Mitosis` is 1.133895

   As we can see from the above list of values, attribute "Mitosis" has the lowest entropy.

   (c) Fill-in the table below with the KL distance for attribute pairs. For this we construct a mass function P_i over A_i by simple counting. For a cell whose row, column entries are A_i, A_j, find $d_{KL}(P_i||P_j)$. You may use an existing `R` function for this, but you need to provide sufficient package details for someone who would consider using that package.

	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9
A_1	0	0.487	0.432	0.594	0.883	0.842	0.495	0.793	4.83
A_2	0.479	0	0.0136	0.0137	1.572	0.152	0.615	0.047	1.848
A_3	0.413	0.014	0	0.034	1.393	0.240	0.491	0.088	2.16
A_4	0.561	0.0132	0.029	0	1.599	0.161	0.606	0.046	1.263
A_5	1.197	1.543	1.313	1.422	0	2.165	0.566	1.883	2.869
A_6	0.747	0.129	0.199	0.155	2.046	0	1.042	0.111	2.689
A_7	0.466	0.664	0.5111	0.644	0.642	1.180	0	0.891	3.973
A_8	0.724	0.045	0.079	0.049	1.959	0.116	0.805	0	3.897
A_9	1.369	0.294	0.354	0.231	2.72	0.306	1.212	0.188	0

The KL distance between attributes of the cancer set.

Solution: First we have to install the package "entropy" which contains "KL.plugin" used to compute the Kullback-Leiber (KL) divergence. It takes the frequencies of each element of the 2 columns(probability distributions of 2 columns) and the unit as arguments and returns the KL divergence between the two columns. The unit is set to 'log2' to compute entropy in bits.
The R code to calculate KL distance is attached as "KL_distance.R". The code imports the "entropy" package, computes the frequencies of each element of each attribute and then uses these frequencies to calculate the KL distance. Using the code, I have updated the above table with the KL distance for each attribute pair.

5. Implement k-means so that you can cluster Δ^*. Since we have labels for the data, we can measure the quality of the clustering. If an element of δ is correctly clustered in c_i (nearest to the correct centroid), then it is considered a True Positive (TP). If an element that correctly belongs to c_i is clustered in a different c_j (in other words, nearer and incorrect centroid), then the element is a False Positive (FP). The Positive Predictive Value (PPV) is

$$PPV = \frac{TP}{TP + FP} \tag{16}$$

Investigate varying the number of blocks as well as the attributes used. There are a modest number of attributes, so should use the powerset. Discuss the result with simply finding pairs of correlations.

Solution: The R code for creating clusters using kmeans and calculating PPVs is attached as "kmeans_PPV_updated.R" file. To display the functionality of each code snippet, I have provided comments wherever necessary. Once the clusters are formed, I calculate the True Positive(TP) and False Positive(FP) values. If a cluster has more number of benign data than malignant data, then total number of benigns in that cluster is TP and total number of malignants is FP and vice versa. I sum up the TP and FP values of k clusters and then calculate the PPV.

The PPV value for each cluster measures the quality of the cluster.
First, I vary the number of blocks(clusters), keeping the attributes to 9 and calculate my PPV values. Below table shows the resultant values -

	PPV
$k = 2$	0.965
$k = 3$	0.963
$k = 4$	0.94
$k = 5$	0.96
$k = 6$	0.96
$k = 7$	0.96
$k = 8$	0.97
$k = 9$	0.95
$k = 10$	0.98

PPVs varying the number of clusters.

As seen from the above table, most of the PPVs are greater than 0.9, indicating that the quality of our clusters are very good even though the number of clusters are varied across the dataset.

Now, we need to find correlation between the attributes and find which are similar to one another. We use the R function "cor" to find the correlation between our attributes - $cor(data)$. The correlation matrix is attached "correlation.xls" file. From the results, we have correlation between -

Uniformity of cell shape and Uniformity of cell size as 0.907

Single Epithelial cell size and Uniformity of cell size as 0.753

Bland Chromatin and Uniformity of cell size as 0.755

I am not considering other values since I have implied a threshold on the correlation value to be ¿0.75. If the value is < 0.75, then they are not very closely related.
The above 3 correlation values suggests that attribute "Uniformity of cell size" is closely related to "Uniformity of cell shape" and even to "Single Epithelial cell size" and "Bland Chromatin".
I first remove the attribute "Uniformity of cell size" and test my algorithm to measure the quality of the clusters. I try this by varying clusters from 2-5. Results are shown below -

	PPV
$k = 2$	0.96
$k = 3$	0.96
$k = 4$	0.935
$k = 5$	0.91

As seen from the above table, most of the PPVs are greater than 0.9, indicating that the quality of our clusters are still very good, after removing the attribute "Uniformity of cell size".
In a similar way, since "Uniformity of cell shape" is also somewhat similar to the other 3 attributes, lets try to measure the quality of the clusters removing the attribute "Uniformity of cell size". Results are shown below -

	PPV
$k = 2$	0.96
$k = 3$	0.92
$k = 4$	0.96
$k = 5$	0.93

Again, the above table suggests that most of the PPVs are greater than 0.9, indicating that the quality of our clusters are still very good, after removing the attribute "Uniformity of cell shape".
I now want to test the quality of my clusters when both the attributes "Uniformity of cell size" and "Uniformity of cell shape" are removed. These both attributes are very closely related to each other as shown in "Correlation_A2_A3.pdf".
Results are shown below -

	PPV
$k=2$	0.96
$k=3$	0.92
$k=4$	0.885
$k=5$	0.95

In the above table, again most of our PPVs are greater than 0.9, indicating that the quality of our clusters are good enough even after removing attributes "Uniformity of cell size" and "Uniformity of cell shape".
In here, I am considering only the attributes whose correlation values are > 0.75. We can consider other pairs of correlation but we might get less quality clusters. This exercise helps us to understand that, removing similar attributes increases the computational speed of the algorithm by a great deal, still maintaining the quality of the cluster.

6. One of the most common techniques in assessing function is using V-fold cross validation. The idea is simple. Suppose $|\Delta^*| = N$. Partition Δ^* into $V = 10$ sets $D^* = \{D_1^*, D_2^*, \ldots, D_{10}^*\}$ such that each $|D_i^*| = \frac{N}{10}$ tuples and all D_i, D_j are pairwise disjoint. The task is to use $V-1$ sets to train and the remaining d to test.
Calculate the PPV using V-fold cross validation. Discuss your results.

Train	Test	PPV Result
kmeans$(D^* - \{D_1^*\})$	D_1^*	0.838
kmeans$(D^* - \{D_2^*\})$	D_2^*	0.94
kmeans$(D^* - \{D_3^*\})$	D_3^*	0.985
kmeans$(D^* - \{D_4^*\})$	D_4^*	0.941
kmeans$(D^* - \{D_5^*\})$	D_5^*	0.926
kmeans$(D^* - \{D_6^*\})$	D_6^*	1
kmeans$(D^* - \{D_7^*\})$	D_7^*	0.955
kmeans$(D^* - \{D_8^*\})$	D_8^*	1
kmeans$(D^* - \{D_9^*\})$	D_9^*	1
kmeans$(D^* - \{D_{10}^*\})$	D_{10}^*	1

Solution: The R code for calculating each PPVs is attached as "kmeans_VFold_updated.R" and the results are updated in the above table.
The total PPV is then

$$PPV(\Delta) \quad = \quad (1/10)\Sigma_{i=1}^{10}\alpha_i PPV(\Delta) = (1/10)9.584 = \mathbf{0.9584} \tag{17}$$

I have run the code for k=2, but this can be extended for any number of clusters.
There is one assumption made in the code. When the 2 clusters are formed initially, whichever cluster has more datasets, I assume that cluster to be "benign" and the other cluster to be "malignant".
This is because in our actual dataset, we have more tuples of data as "benign", with label as 2(458 out of 699).
Another reason for my assumption is, if we classify the centroid as "benign" when the number of datasets with label as 2 is more, we might end up with both clusters as "benign". This is not wrong, but in order to have a better visualization of D_1^* and to make better sense of distance function, I have made this assumption. (If both clusters are benign, then distance function won't make much sense while assigning individual data points of D_1^*).

V-fold cross validation measures the quality of our clusters when we apply our test dataset on the training results. It is a useful technique to overcome the problem of over-fitting. We need to make sure that our

algorithm works well for both training and test dataset and the table results denote the same.

For some test datasets such as D_6^*, D_8^*, D_9^* and D_10^*, PPV value of 1 indicates that these datasets are clustered with 100% accuracy. The other datasets also have PPVs close to 0.9 indicating that the clustering is 90% accurate.
The total PPV value of 0.9584 indicates that the quality of the clusters formed is good enough to run any test data and is confirmed by V-fold cross validation.

NOTE: I have uploaded all the R files and the source file used for computations, "meddata_na.csv". These files need to be copied to the working directory before executing.