CS-734/834 Introduction to Information Retrieval: Assignment #4: Ex 8.3, 8.4, 9.4, 9.8 & 9.10

Due on Thursday, December 8, 2016

Dr. Michael L. Nelson

Plinio Vargas
pvargas@cs.odu.edu

Contents

List of Figures

Listings

List of Tables

1 Exercise 8.3

For one query in the CACM collection (provided at the book website), generate a ranking using Galago, and then calculate average precision, NDCG at 5 and 10, precision at 10, and the reciprocal rank by hand.

1.1 Approach

The CACM collection query log has a great deal of queries with a large number of words. This makes the query results in Galago somewhat difficult to interpret without some type of description of what should be considered as a relevant result. For example, the TREC query log contains a narrative of what to consider in order to mark a document relevant. This is not the case for the CACM collection. Below, it is an example of the narrative found in the TREC collection.

```
<narr> Narrative:
Relevant documents must include details of how pet! or animal!assisted
therapy is or has been used.  Relevant details include information
about pet therapy programs, descriptions of the circumstances in which
pet therapy is used, the benefits of this type of therapy, the degree
of success of this therapy, and any laws or regulations governing it.
```

The 12^{th} query in the log was used for this exercise:

```
</DOC>
<DOC>
<DOCNO> 12 </DOCNO>
 portable operating systems
</DOC>
```

A scale of 0 through 3 was used. Where 3 is the most relevant and 0 is not relevant. The following was the criteria used to determined the documents query result relevance:

- If the document is related to "portable operating system" at 3 points was awarded
- If the document refers to portable software 2 points were awarded
- If the document is related to operating system at least 1 point will be awarded
- The occurrence of the word portable unrelated to operating system provided no relevance to the query

1.2 Solution

After placing "portable operating system" in **Galago** the 10 first results were analyzed for relevance.

Exercise 8.3 continued on next page...


localhost:43066/search?q=portable+operating+systems

GALAGO

portable operating systems Search

[debug]

CACM-3127
Thoth, a **Portable** Real-Time **Operating** System Thoth isa real-time **operating** system which is designed to be **portable** over a large set...which use Thoth are highly **portable**. Thoth encourages structuring programs...Sager, G. Portability, real time, **operating** systems, minicomputer 3.80 4.30...
CACM-3127 - null

CACM-2246
Levels of Language for **Portable** Software An increasing amount...is being implemented in a **portable** form. A popular way...December, 1972 Brown, P. J. **portable** software, level of language...
CACM-2246 - null

CACM-1930
Extremely **Portable** Random Number Generator Extremely **portable** subroutines are sometimes needed...of this sort. An extremely **portable** 8-line FORTRAN program...
CACM-1930 - null

CACM-3196
...The reactive typewriter should be **portable**. the reactive typewriter should
CACM-3196 - null

CACM-2593
CACM-2593 - null

CACM-1591
...that was incorporated into an **operating** system of a large...model transferred control to the **operating** system to execute functions...written to run under the **operating** system (IBSYS) for the...the full resources of the **operating** system (language processors, compilers...
CACM-1591 - null

CACM-1680
...IBM 2250 display Unit under **Operating** System/360. Adept is...program that controls the standard **operating** system by terminating and...and surrendering control to the **operating** system to perform other...system-cataloged programs) of the **operating** system. Language processors and...L. computer-assisted instruction, tutorial **systems**, programming, simulation, modeling, information...
CACM-1680 - null

CACM-2740
A Large Semaphore Based **Operating** System The paper describes...internal structure of a large **operating** system as a set...to Dijkstra's hierarchical structuring of **operating systems**. The project management and...performance are discussed, too. The **operating** system is the

1.2.1 Generated Ranking Using Galago

Using the relevance criteria mentioned before the following scores were given to the ten retrieved documents:

Table 1: Ranking for Galago Query #1

Document	Ranking	Relevance Score
CACM-3127	1	3
CACM-2246	2	2
CACM-1930	3	0
CACM-3196	4	0
CACM-2593	5	0
CACM-1591	6	2
CACM-1680	7	1
CACM-2740	8	2
CACM-3068	9	1
CACM-2319	10	2

Query "portable operating system" in Galago using CACM collection.

1.2.2 Calculating Recall and Precision

Recall Calculation

By looking at Table 1, the *recall* calculation of the i^{th} document d resulting from **Query#1** as follows:

$$r_i = \frac{\sum_{k=1}^{i} d_k \begin{cases} d_k = 1 \; if \; d_k \; is \; relevant \; 0 \; otherwise \end{cases}}{|Query1|} \tag{1}$$

$|Query1|$ =total number of document return in Query#1.

In other words, the *recall* calculation r_i of the i^{th} ranked document d resulted from Query#1 is going to be equal to the count of relevant documents retrieved from the 1^{st} ranked to the i^{th} ranked, divided by the total number of relevant documents related to the query.

Precision Calculation

By looking at Table 1, the *precision* calculation of the i^{th} document d resulting from **Query#1** as follows:

$$p_i = \frac{\sum_{k=1}^{i} d_k \begin{cases} d_k = 1 \; if \; d_k \; is \; relevant \; 0 \; otherwise \end{cases}}{i} \tag{2}$$

In other words, the *precision* calculation p_i of the i^{th} ranked document d resulted from Query#1 is going to be equal to the count of relevant documents retrieved from the 1^{st} ranked to the i^{th} ranked, divided by the rank of the i^{th} document related to the query.

By applying equations (1) and (2) to the values of Table 1, it resulted in the *recall* and *precision* calculation shown on Table 2.

Summary

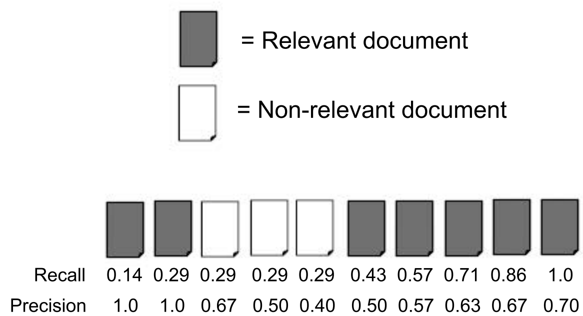


Table 2: Precision - Recall Calculation at 10 for Query #1

i^{th} Rank	Is Relevant	Recall	Precision
1	Yes	0.14	1.00
2	Yes	0.29	1.00
3	No	0.29	0.67
4	No	0.29	0.50
5	No	0.29	0.40
6	Yes	0.43	0.50
7	Yes	0.57	0.57
8	Yes	0.71	0.63
9	Yes	0.86	0.67
10	Yes	1.00	0.70

Query “portable operating system” in Galago using CACM collection.

1.2.3 Average Precision at 10

From Table 2 the Average Precision for Query #1 (APQ1) is:

$$APQ1 = (1.0 + 1.0 + 0.57 + 0.63 + 0.67 + 0.70)/6$$

$$APQ1 = 0.72$$

1.2.4 NDCG at 5

Table 3: DCG calculation at 5 for Query #1

i^{th} Rank	Assigned Relevance	LOG(i)	Discounted Gain	DCG
1	3	0.00	3.00	3.00
2	2	1.00	2.00	5.00
3	0	1.58	0.00	5.00
4	0	2.00	0.00	5.00
5	0	2.32	0.00	5.00

Query "portable operating system" in Galago using CACM collection. Discounted gain is calculated for the 1^{st} ranked document by the"Assigned Relevance" value. The remaining i^{th} documents are calculated by dividing **Assigned Relevance** by LOG(i)

Table 4: NDCG calculation at 5 for Query #1

i^{th} Rank	Perfect Ranking	LOG(i)	Discounted DCG	Ideal DCG	DCG	NDCG
1	3	0.00	3.00	3.00	3.00	1.00
2	2	1.00	2.00	5.00	5.00	1.00
3	2	1.58	1.26	6.26	5.00	0.80
4	1	2.00	0.50	6.76	5.00	0.74
5	0	2.32	0.00	6.76	5.00	0.74

Query "portable operating system" in Galago using CACM collection. Discounted gain is calculated for the 1^{st} ranked document by the"Assigned Relevance" value. The remaining i^{th} documents are calculated by dividing **Assigned Relevance** by LOG(i)

Looking at Table 4:

NDCG at 5 = 0.74

1.2.5 NDCG at 10

Table 5: DCG calculation at 10 for Query #1

i^{th} Rank	Assigned Relevance	LOG(i)	Discounted Gain	DCG
1	3	0.00	3.00	3.00
2	2	1.00	2.00	5.00
3	0	1.58	0.00	5.00
4	0	2.00	0.00	5.00
5	0	2.32	0.00	5.00
6	2	2.58	0.77	5.77
7	1	2.81	0.36	6.13
8	2	3.00	0.67	6.80
9	1	3.17	0.32	7.11
10	2	3.32	0.60	7.71

Query "portable operating system" in Galago using CACM collection. Discounted gain is calculated for the 1^{st} ranked document by the"Assigned Relevance" value. The remaining i^{th} documents are calculated by dividing **Assigned Relevance** by LOG(i)

Table 6: NDCG calculation at 10 for Query #1

i^{th} Rank	Perfect Ranking	LOG(i)	Discounted DCG	Ideal DCG	DCG	NDCG
1	3	0.00	3.00	3.00	3.00	1.00
2	3	1.00	3.00	6.00	5.00	0.83
3	3	1.58	1.89	7.89	5.00	0.63
4	2	2.00	1.00	8.89	5.00	0.56
5	2	2.32	0.86	9.75	5.00	0.51
6	2	2.58	0.77	10.53	5.77	0.55
7	1	2.81	0.36	10.88	6.16	0.56
8	0	3.00	0.00	10.88	6.80	0.62
9	0	3.17	0.00	10.88	7.11	0.65
10	0	3.32	0.00	10.88	7.71	0.71

Query "portable operating system" in Galago using CACM collection. Discounted gain is calculated for the 1^{st} ranked document by the"Assigned Relevance" value. The remaining i^{th} documents are calculated by dividing **Assigned Relevance** by LOG(i)

Looking at Table 6:

NDCG at 10 = 0.71

1.2.6 Reciprocal Ranking

According to [1] *reciprocal rank* "is defined as the reciprocal of the rank at which the first relevant document is retrieved", then given that the first document of the query result is a relevant document, the relevant ranking RR can be determined by:

$$RR = 1/1 = 1$$

2 Exercise 8.4

For two queries in the CACM collection, generate two uninterpolated recall-precision graphs, a table of interpolated precision values at standard recall levels, and the average interpolated recall-precision graph.

2.1 Approach

In addition to query #1, used on previous exercise, the second query is shown below:

```
<DOC>
<DOCNO> 2 </DOCNO>
I am interested in articles written either by Prieve or Udo Pooch
Prieve, B.
Pooch, U.
</DOC>
```

A scale of 0 through 3 was used. Where 3 is the most relevant and 0 is not relevant. The following was the criteria used to determined the documents query result relevance:

- If any authors created the document 3 points was awarded
- If any of the authors appeared in the body of the document 2 points were awarded
- If the document was related to any of the writing of the authors 1 point was awarded
- Any other condition 0 was awarded

2.2 Solution

After placing the query #2 in **Galago**, the 10 first results were analyzed for relevance.

The NDCG at 10 calculation for Query #2 generated the following results:

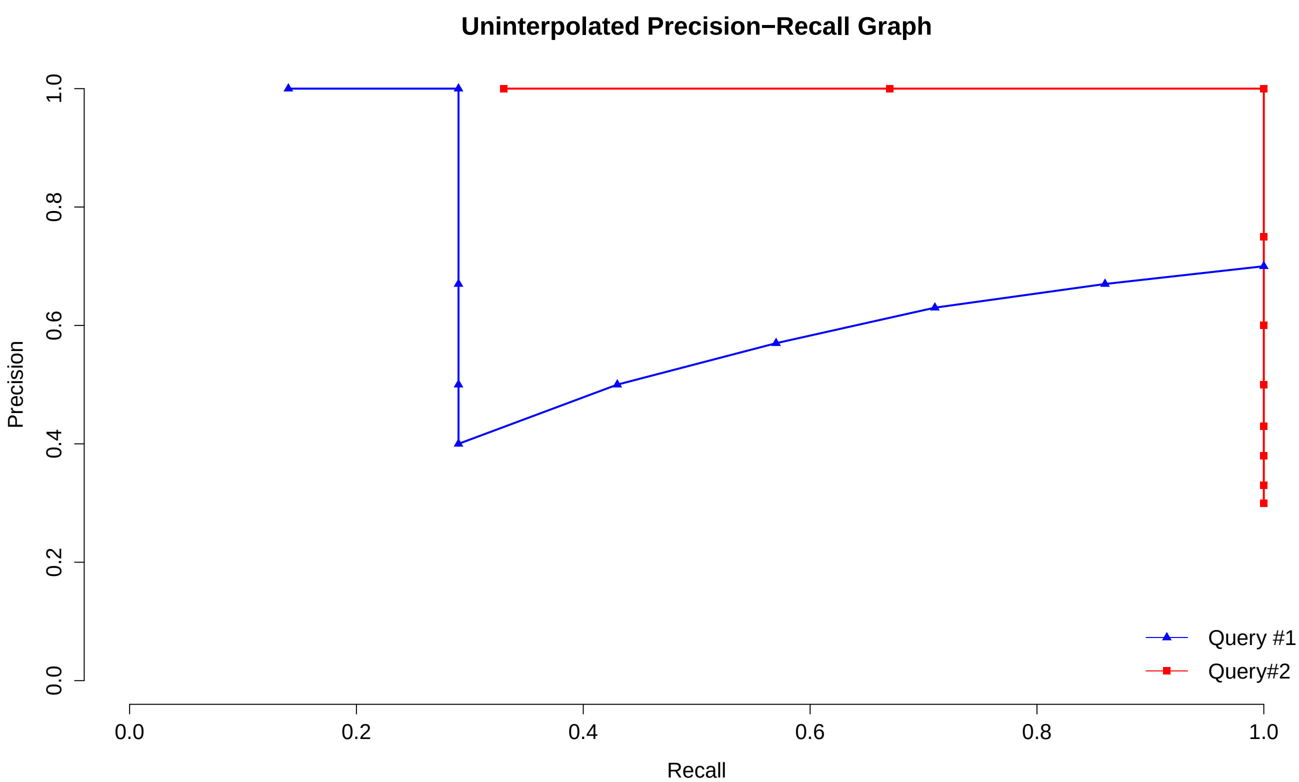
Table 7: Precision - Recall Calculation at 10 for Query #2

i^{th} Rank	Is Relevant	Recall	Precision
1	Yes	0.33	1.00
2	Yes	0.67	1.00
3	Yes	1.00	1.00
4	No	1.00	0.75
5	No	1.00	0.60
6	No	1.00	0.50
7	No	1.00	0.43
8	No	1.00	0.38
9	No	1.00	0.33
10	No	1.00	0.30

Query "I am interested in articles written either by Prieve or Udo Pooch" in Galago using CACM collection.

2.2.1 Uninterpolated Recall Precision Graph

The **R** script *query.R* was used to generate the graph. The data for this graph can be seen on Table 8.



2.2.2 Table of Interpolated Precision Values

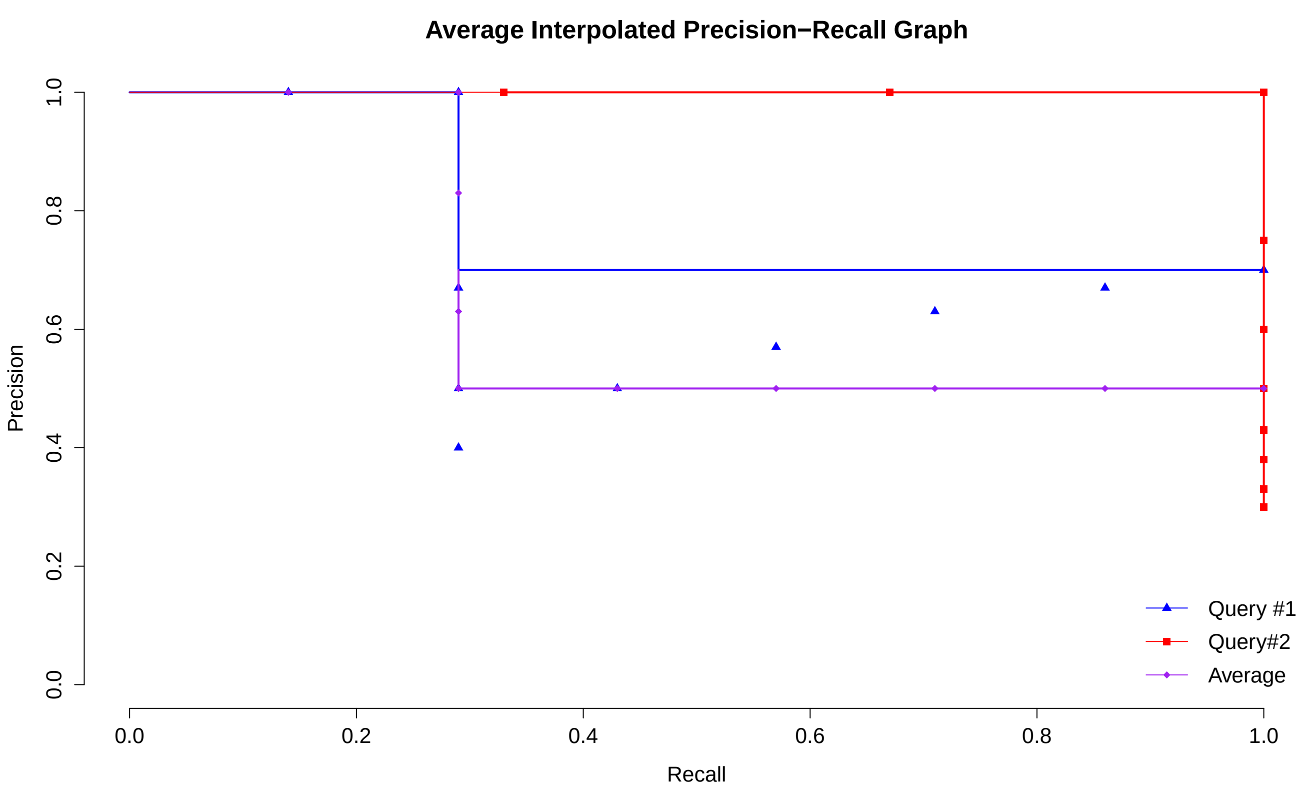
Table 8: Interpolated Recall-Precision Values for Query 1 and 2

Recall	0.14	0.29	0.29	0.29	0.29	0.43	0.57	0.71	0.86	1.0
Ranking 1	1.00	1.00	0.67	0.50	0.40	0.50	0.57	0.63	0.67	0.70
Ranking 2	1.00	1.00	1.00	0.75	0.60	0.50	0.43	0.38	0.33	0.30
Average	1.00	1.00	0.83	0.63	0.50	0.50	0.50	0.50	0.50	0.50

Query 1 and 2

2.2.3 Average Interpolated Recall-Precision Graph

The **R** script *avg-precision.R* was used to generate the graph. The data for this graph can be seen on Table 8.



3 Exercise 9.4

For some classification data set, compute estimates for $P(w|c)$ for all words w using both the multiple-Bernoulli and multinomial models. Compare the multiple-Bernoulli estimates with the multinomial estimates. How do they differ? Do the estimates diverge more for certain types of terms?

3.1 Approach

The CACM collection was used to obtain the data set. No stop-words were used. In order to eliminate the stop-words the application from assignment #2, *zipf_curve.py*, was used to determine the words containing the higher frequency. Then, the next 20 highest frequency terms were selected, without including any numbers or non-relevant words which may be part of the index.

3.1.1 Data set Classification

Considering that the CACM collection is related to computer science, the classification selection for this exercise were: **hardware** and **software**.

3.1.2 Word list

Table 9 contains the list of the words used for this exercise. An evaluation was made using **Galago** to determine 20 documents relevant to **hardware** as the set to classify the term appearance probability for which our word list is present. Similar approach was used to obtain 20 documents relevant to the classification for **software**.

Exercise 9.4 continued on next page...

Table 9: Word Selection to Compute *P(w|c)* Estimates

i	Word	Freq
1	algorithm	1617
2	computer	1131
3	data	929
4	program	818
5	language	751
6	method	720
7	systems	692
8	information	498
9	problem	489
10	number	464
11	processing	410
12	memory	395
13	design	353
14	function	335
15	structure	294
16	languages	290
17	methods	289
18	model	284
19	matrix	272
20	control	269

Words were selected from the CAMC collection taking in consideration non-stopwords with high frequency occurrence in the collection

3.1.3 Document Classification Criteria

Snippet information was used to consider if it was worthy to inspect the document for classification selection. If a document appeared to be related to software and hardware, it was discarded in order to have a good representation of the classified document.

3.1.4 Generating the Data

The **Python** script *bernoulli-estimate.py* was developed to generate the data contained on Tables 10 and 11. The script read two files: *hardware.txt* and *software.txt*. Those files contain the documents in the collection considered related to a **hardware** and **software** query respectively for the training. A dictionary variable *word_matrix* was passed as a parameter (line 178 of Listing 1) where the feature frequency is going to be stored.

If a training document corresponded to a *hardware* classification, then a column with a value of 'H' was added to the row. This action was the only step needed to training the document instance as *hardware*. The same was done for the *software* training document classification. See lines 215-218

Listing 1: Frequency of Term Count

```
178 def count_words(url, words, word_matrix):
179     # get hardware training file
180     hardware = []
181     with open('hardware.txt', 'r') as f:
```

Exercise 9.4 continued on next page...

```
        for line in f:
            hardware.append(line.strip() + '.html')

    software = []
    with open('software.txt', 'r') as f:
        for line in f:
            software.append(line.strip() + '.html')

    for filename in os.listdir(url):
        if filename in hardware or filename in software:
            f = open(os.path.join(url, filename), 'r')
            page = f.read()
            f.close()

            print(url)
            soup = BeautifulSoup(page, 'html.parser')
            data = soup.get_text()
            data = re.sub('[*#/=?&>}{!<)(;,|\"\.\[\]]', ' ', data)


            # find all features and increment frequency
            row = [0 for x in range(len(words))]
            for unigram in data.split():
                unigram = unigram.lower()

                # remove empty string
                if len(unigram) > 0 and unigram != 's' and unigram != '-' and \
                        unigram != '' and unigram != '' and unigram != '--' and \
                        unigram != ' ':
                    unigram = unigram.strip("'")
                    if unigram in words:
                        row[words[unigram]] += 1

            if filename in hardware:
                row.append('H')
            else:
                row.append('S')

            word_matrix.append(row)
            del data, page, soup

    return
```

Two functions were created to make the distinction while considering the presence of a term, used in the **multiple-bernoulli** method, or the frequency of the term, used in the *multinomial* method. Looking at lines 152 through 155 on Listing listing:bernoulli-fucntion, we can notice the function removed the frequency and converted any value greater than zero equal to 1. The **multinomial** function accomplished the same job, but it considered the frequency of the term instead of only its presence.

Listing 2: Multiple-Bernoulli Function

```
139 def bernoulli_multiple(matrix):
140     n = len(matrix[0]) - 1  # number of term/words in training matrix
```

```
    # initialize c1 class vector
    df_c1 = [0 for x in range(n)]

    # initialize c2 class vector
    df_c2 = [0 for x in range(n)]

    # link classifier to rows
    classifier = {'H': df_c1, 'S': df_c2}
    class_size = {'H':0, 'S': 0}

    for row in matrix:
        fix_row = [x if x > 0 else 1 for x in row[:n]]    # fix div by 0 problem
        classifier[row[n]] = np.sum([np.floor_divide(row[:n], fix_row), classifier
            [row[n]]], axis=0)
        class_size[row[n]] += 1

    return classifier, class_size
```


hardware - Galago Search
localhost:39556/search?q=hardware
GALAGO
hardware
Search
CACM-2967
A Comparison of Hardware and Software Associative Memories...Drawings (APLD) System utilizes a hardware associative memory and creates...Structure, A comparison of the hardware approach with the software...illustrates the advantages of the hardware associative memory in three...
CACM-2967 - null
CACM-2377
A Hardware Architecture for Implementing Protection...a computation. This paper describes hardware processor mechanisms for implementing...trapping to the supervisor. Automatic hardware validation of references across...H. protection, protection rings, protection hardware, access control, hardware access control, computer utility...
CACM-2377 - null
CACM-2424
...on the decision using independent hardware and software. The dynamic...the presence of a single hardware or software fault. Furthermore...the presence of a single hardware or software fault.the amount of additional hardware and software required for...
CACM-2424 - null
CACM-3025
...origin and evolution of the hardware, operating system, and languages...time sharing computing systems; transferring hardware technology within DEC (and...design and manufacturing; supporting minicomputer hardware and software development; and...
CACM-3025 - null
CACM-2277
...on the MANIAC II A hardware implementation on the Maniac...added to the Maniac II hardware. Finally, a description of the hardware design for implementation of...
CACM-2277 - null
CACM-2928
Hardware Estimation of a Process' Primary Memory Requirements A minor hardware extension to the Honeywell...to be approximated. The additional hardware required for this estimate...
CACM-2928 - null
CACM-2625
...identify capabilities will dominate. A hardware address translation scheme which...R. S. addressing, capabilities, addressing hardware, protection, protection hardware, shared addresses, information sharing...
CACM-2625 - null
CACM-0595
...that some knowledge of the hardware and computer logic must...a student must know the

3.1.5 Multiple-Bernoulli Model

$P(w|c)$ for all words on Table 9, using **multiple-Bernoulli** model, was calculated by applying the formula:

$$P(w|c) = \frac{df_{w,c} + 1}{N_c + 1} \tag{3}$$

Where $df_{w,c}$ refers to the number of training documents with class label c in which term w occurs, and N_c is the total number of training documents with class label c. Knowing that 20 training documents were used for each of the classification in this exercise:

$$N_{hardware} = 20$$

and,

$$N_{software} = 20$$

3.1.6 Multinomial Model

$P(w|c)$ for all words on Table 9, using **multinomial** model, was calculated by applying the formula:

$$P(w|c) = \frac{tf_{w,c} + 1}{|c| + |\mathcal{V}|} \tag{4}$$

Where $tf_{w,c}$ refers to the number of times that term w occurs in document d, $|c|$ is the total number of terms that occur in training documents with class label c. $|\mathcal{V}|$ is the number of features (20 words in our case) considered to classify the document. After scanning the collection for the selected training document, the following $|c|$ values resulted to provide inputs for equation (4):

$$|c_{hardware}| = 190$$

and,

$$|c_{software}| = 137$$

Exercise 9.4 continued on next page...

3.2 Solution

3.2.1 Multiple-Bernoulli Model

Table 10: Multiple-Bernoulli Model Document Representation

	algorithm	computer	data	program	language	method	systems	information	problem	number	processing	memory	design	function	structure	languages	methods	model	matrix	processing	
1	0	1	1	0	1	0	0	1	1	0	1	1	0	0	1	0	0	0	0	0	Hardware
2	0	0	1	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0	Software
3	0	1	0	0	1	0	0	0	1	0	0	0	1	0	0	1	0	0	0	1	Software
4	0	0	1	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	Software
5	0	0	1	0	0	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	Hardware
6	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Hardware
7	0	1	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	Software
8	0	1	1	0	0	0	1	0	0	0	1	0	1	0	0	0	0	1	0	0	Hardware
9	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	Hardware
10	0	1	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	1	Hardware
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Software
12	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Software
13	0	0	0	0	0	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	Software
14	0	1	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	Hardware
15	0	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	Software
16	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Hardware
17	0	1	1	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0	0	1	Hardware
18	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	Software
19	0	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	Hardware
20	0	0	0	1	1	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	Software
21	0	0	1	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	Software
22	0	1	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	Hardware
23	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	Software
24	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	Software
25	0	1	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	0	Software
26	0	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	0	Hardware
27	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	Hardware
28	0	1	1	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	Software
29	0	1	0	0	0	0	1	1	0	0	0	1	1	0	1	0	0	0	0	1	Hardware
30	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	Software
31	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Software
32	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	Hardware
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Software
34	0	1	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	Hardware
35	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	Hardware
36	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	1	0	0	0	Hardware
37	0	1	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	Hardware
38	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	Software
39	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	Hardware
40	0	0	1	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	Software
	1	13	8	2	1	3	10	6	3	1	2	8	10	2	5	1	2	4	0	4	Hardware
	0	7	7	8	4	0	10	3	3	1	2	3	11	1	3	1	0	2	0	2	Software
	0.048	0.381	0.381	0.429	0.238	0.048	0.524	0.190	0.190	0.095	0.143	0.190	0.571	0.095	0.190	0.095	0.048	0.143	0.048	0.143	Software
	0.095	0.667	0.429	0.143	0.095	0.190	0.524	0.333	0.190	0.095	0.143	0.429	0.524	0.143	0.286	0.095	0.143	0.238	0.048	0.238	Hardware

Words were selected from the CAMC collection taking in consideration non-stopwords with high frequency occurrence in the collection

3.2.2 Multinomial Model

Table 11: Multinomial Model Document Representation

	algorithm	computer	data	program	language	method	systems	information	problem	number	processing	memory	design	function	structure	languages	methods	model	matrix	processing	
1	0	2	2	0	1	0	0	1	1	0	3	5	0	0	1	0	0	0	0	0	Hardware
2	0	0	1	0	0	0	2	0	0	0	0	1	2	0	0	0	0	0	0	0	Software
3	0	2	0	0	6	0	0	0	1	0	0	0	7	0	0	2	0	0	0	1	Software
4	0	0	3	3	0	0	0	1	0	0	1	0	0	0	1	0	0	1	0	0	Software
5	0	0	1	0	0	0	3	6	0	0	0	6	2	0	0	0	0	0	0	0	Hardware
6	0	2	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	Hardware
7	0	4	0	2	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	Software
8	0	2	2	0	0	0	3	0	0	0	2	0	5	0	0	0	0	6	0	0	Hardware
9	0	0	0	0	0	0	4	0	0	0	0	0	1	0	0	0	1	0	0	0	Hardware
10	0	2	0	0	0	0	0	2	0	0	0	2	0	0	0	0	0	0	0	2	Hardware
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Software
12	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	Software
13	0	0	0	0	0	0	1	0	1	0	0	0	2	0	0	0	0	0	0	0	Software
14	0	2	3	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	Hardware
15	0	0	0	5	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	Software
16	0	0	1	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Hardware
17	0	4	1	0	0	0	3	0	0	0	0	0	0	0	1	1	0	0	0	1	Hardware
18	0	0	7	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	Software
19	0	0	1	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	1	Hardware
20	0	0	0	1	3	0	1	1	0	0	1	0	4	0	0	0	0	0	0	0	Software
21	0	0	3	0	5	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	Software
22	0	2	0	0	0	0	3	1	0	0	0	2	0	0	0	0	0	0	0	0	Hardware
23	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	Software
24	0	0	0	0	0	0	1	2	0	0	0	0	1	0	0	0	0	0	0	0	Software
25	0	2	0	3	0	0	1	0	0	0	0	4	0	0	3	0	0	0	0	0	Software
26	0	0	0	0	0	0	5	0	0	1	0	0	2	0	0	0	0	3	0	0	Hardware
27	0	5	0	0	0	0	3	0	0	0	0	0	0	1	0	0	0	0	0	0	Hardware
28	0	1	1	0	0	0	1	0	0	0	0	0	3	0	0	0	0	0	0	0	Software
29	0	4	0	0	0	0	1	1	0	0	0	4	3	0	1	0	0	0	0	1	Hardware
30	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	Software
31	0	2	0	2	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	Software
32	0	1	0	0	0	0	0	0	0	0	0	4	1	0	0	0	0	4	0	0	Hardware
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	Software
34	0	2	0	5	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	Hardware
35	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	Hardware
36	0	0	0	0	0	1	0	0	2	0	0	0	2	1	0	0	1	0	0	0	Hardware
37	0	1	5	0	0	0	0	3	0	0	0	4	0	0	0	0	0	0	0	0	Hardware
38	0	0	0	0	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	Software
39	1	0	0	0	0	0	0	0	0	0	0	1	1	0	2	0	0	0	0	0	Hardware
40	0	0	1	3	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	Software
190	1	30	16	6	1	4	28	14	5	1	5	28	20	2	6	1	2	15	0	5	Hardware
137	0	13	17	20	15	0	16	4	3	2	2	6	25	2	5	2	0	2	0	3	Software
	0.010	0.148	0.081	0.033	0.010	0.024	0.138	0.071	0.029	0.010	0.029	0.138	0.100	0.014	0.033	0.010	0.014	0.076	0.005	0.029	Hardware
	0.006	0.089	0.115	0.134	0.102	0.006	0.108	0.032	0.025	0.019	0.019	0.045	0.166	0.019	0.038	0.019	0.006	0.019	0.006	0.025	Software

Words were selected from the CAMC collection taking in consideration non-stopwords with high frequency occurrence in the collection

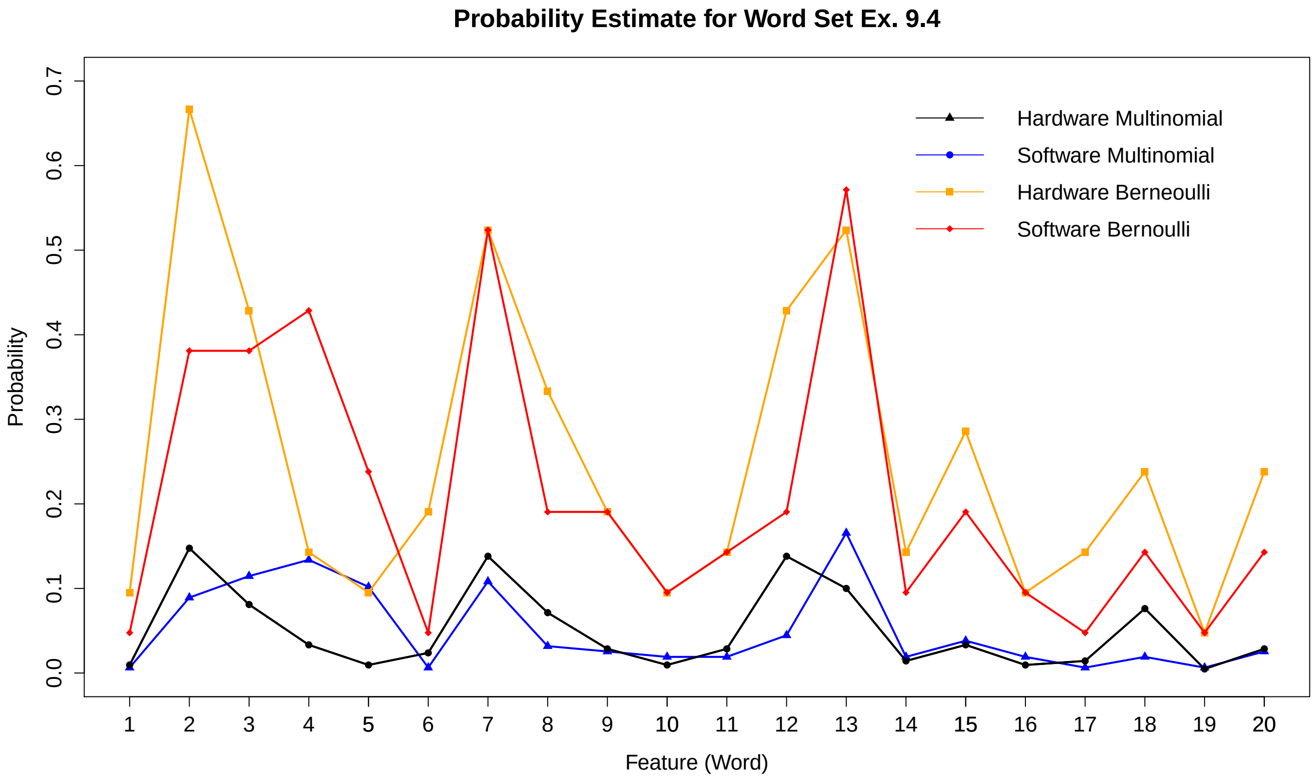
Exercise 9.4 continued on next page...

3.2.3 Multiple-Bernoulli vs Multinomial Estimates Comparison

Table 10 and Table 11 yielded the data to the plot on Fig 1,, the probability estimates for the *Multiple-Bernoulli* and *Mutinomial* model. As equation(4) suggests, the probabilities estimates in the *Multinomial* model are going to generate a much smaller value, since in comparison with the *Bernoulli* model, the denominator in equation (4) is greater than in equation (1).
Although, here we are not trying to calculate the *information gain* of the features for this data set, the graph can give us some information which can provide us with some insight. For example, there are various points in the graph which pairs intercept. In other words, both variables have the same probability estimate. The result for the *Bernoulli* estimate (red and orange curves) shows that variables $\{7, 9, 10, 11, 16, 19\}$ have the same values. The values of those features are: {system, problem, number, processing, languages, matrix} respectively.
In the other hand, the *Multinomial* model result does not have a single value which probability estimate is exactly the same. There are some instances where the probability appear to converge, but there still a very small percentage where a distinction could be made. According to the textbook [1]: "in practice, the multinomial model has been shown to consistently outperform the multiple-Bernoulli model", perhaps the reason for that is that we can make use of more features in the *multinomial* model, in contrast with the *multiple-Bernoulli* where in our case the quantity of variable with equal estimate will make more difficult to make an accurate document classification.
However, about the same number of features (in our case the vairables $\{1, 9, 14, 19, 20\}$) in the *multinomial* model had an estimate delta close to zero. A common denominator in these two sets are $\{9, 19\}$ (problem and languages). It makes prefect sense that these two words do not appear add more information to distinguish or classify if a document is related to **hardware** or **software**. Looking at Table 10, the word *matrix* does not appear in any of the searched documents, and *problem* is such a generic word that probably could be use in the ACMC collection.

The **R** script used to create Fig 1 was *probability.R*

Figure 1: Probability Estimate



4 Exercise 9.8

Cluster the following set of two-dimensional instances into three clusters using each of the five agglomerative clustering methods:
(4, 2), (3, 2), (2, 2), (1, 2), (1, 1), (1, 1), (2, 3), (3, 2), (3, 4), (4, 3)
Discuss the differences in the clusters across methods. Which methods produce the same clusters? How do these clusters compare to how you would manually cluster the points?

4.1 Approach

R includes the package *cluster* which contains a solution to the five agglomerative cluster methods: *Single linkage*, *Complete linkage*, *Average linkage*, *Average group linkage* and *Wards method*. The name of the script that plotted the cluster is *agnes.R*. The package has a function (*agnes*) that produces a dendrogram of the clustered points.

In order to know how to group the dendrogram in k number of cluster, **R** provides the function *cutree* which divides the graph into k partitions. A portion of the **R** scripts that performs this function is shown below:

```
groups <- cutree(ag, k=5) # cut tree into 3 clusters
rect.hclust(H.fit, k=5, border="red")
```

4.2 Solution

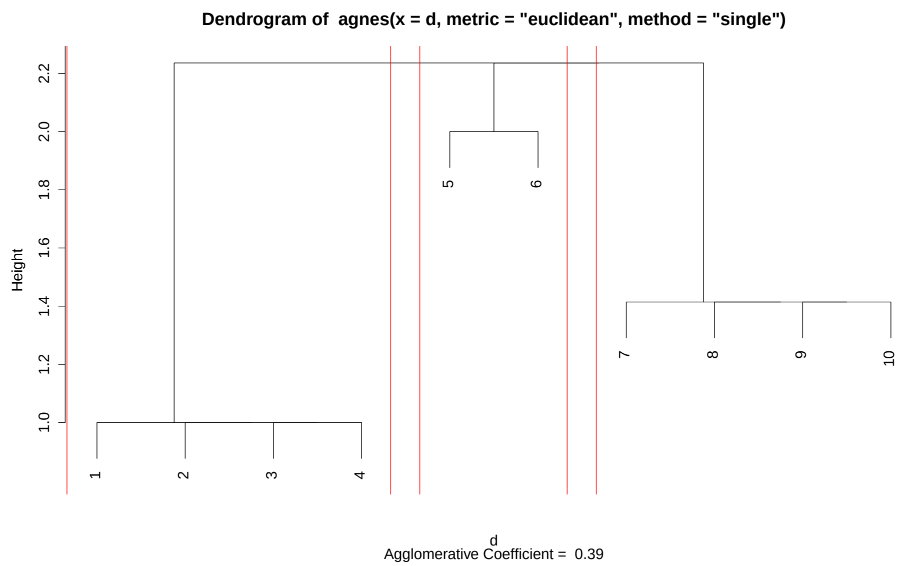
4.2.1 Single Linkage Method

For this method **R** produced the following result:

```
Call:  agnes(x = d, metric = "euclidean", method = "single")
Agglomerative coefficient:  0.3892469
Order of objects:
 [1]  1  2  3  4  5  6  7  8  9 10
Height (summary):
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.000   1.414   1.524   2.000   2.236

Available components:
[1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method"
[8] "data"
```

Figure 2: Single Linkage Graph K=3



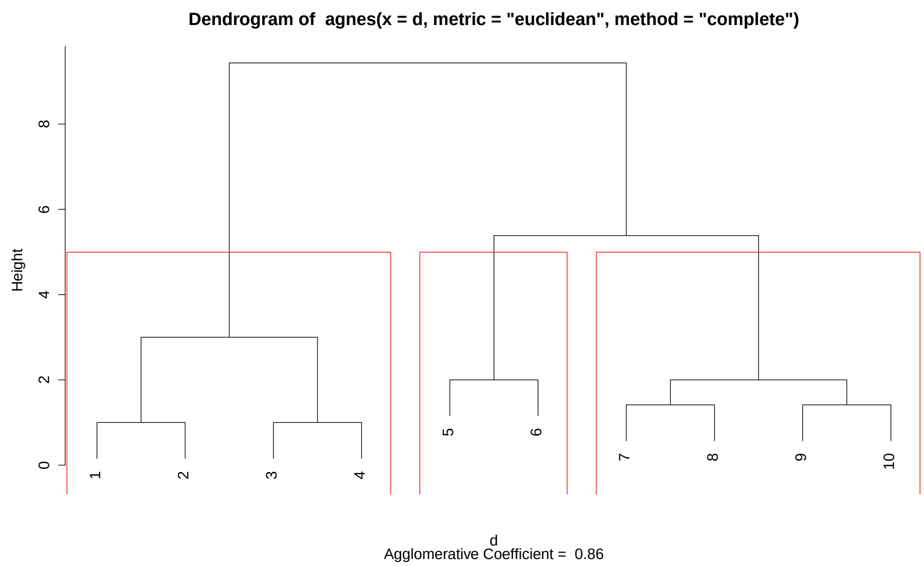
4.2.2 Complete linkage Method

For this method **R** produced the following result:

```
Call:  agnes(x = d, metric = "euclidean", method = "complete")
Agglomerative coefficient:  0.8552376
Order of objects:
 [1]  1  2  3  4  5  6  7  8  9 10
Height (summary):
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.414   2.000   2.961   3.000   9.434

Available components:
[1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method"
[8] "data"
```

Figure 3: Complete Linkage Graph K=3



4.2.3 Average linkage

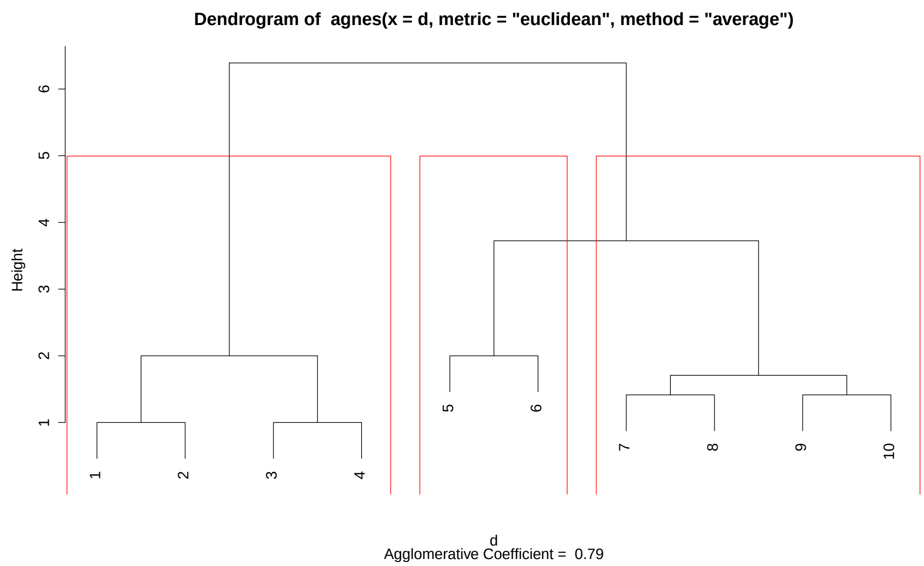
For this method **R** produced the following result:

```
Call:  agnes(x = d, metric = "euclidean", method = "average")
Agglomerative coefficient:  0.7863225
Order of objects:
 [1]  1  2  3  4  5  6  7  8  9 10
Height (summary):
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.414   1.707   2.295   2.000   6.391

Available components:
[1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method"
[8] "data"
```

Exercise 9.8 continued on next page...

Figure 4: Average Linkage Graph K=3



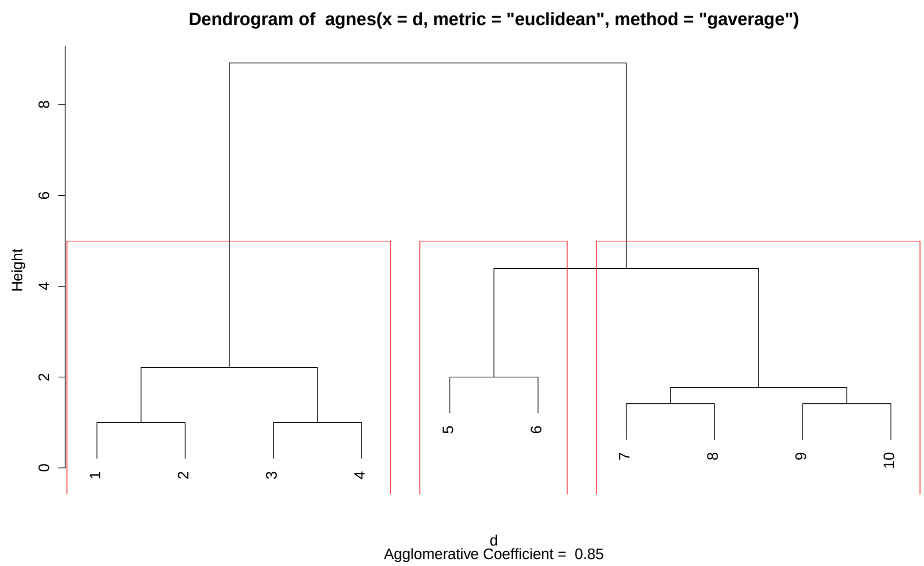
4.2.4 Average Group Linkage

For this method **R** produced the following result:

```
Call:  agnes(x = d, metric = "euclidean", method = "gaverage")
Agglomerative coefficient:  0.8468464
Order of objects:
 [1]  1  2  3  4  5  6  7  8  9 10
Height (summary):
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.414   1.769   2.680   2.210   8.917

Available components:
[1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method"
```

Figure 5: Average Group Linkage Graph K=3



4.2.5 Wards method

For this method **R** produced the following result:

```
Call:  agnes(x = d, metric = "euclidean", method = "ward")
Agglomerative coefficient:  0.9006435
Order of objects:
 [1]  1  2  3  4  5  6  7  8  9 10
Height (summary):
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.414   2.000   3.477   2.828  13.750

Available components:
[1] "order"  "height" "ac"     "merge"  "diss"   "call"   "method"
[8] "data"
```

4.2.6 Comparison of All Agglomerative Methods

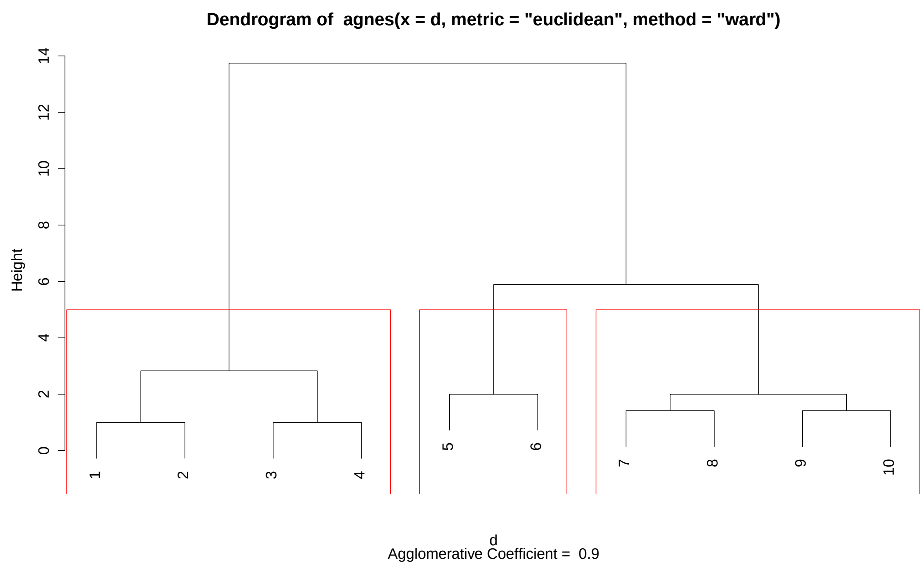
All five methods resulted in similar cluster of two-dimensional set instances with a $k = 3$. However, they differ in the *agglomerative coefficient* which indicates the number of observations required to cluster the set.

The *group average linkage* and *complete linkage* have similar agglomerative coefficient, **0.845** and **0.855** respectively. Looking at their graphs, they appear to be very similar.

The *Single Linkage* has the smallest agglomerative coefficient *0.39*.

Exercise 9.8 continued on next page. . .

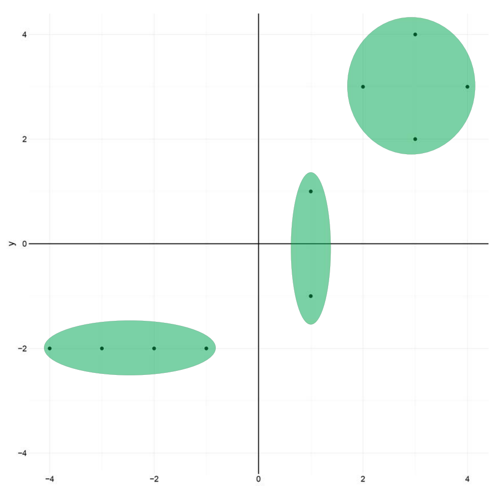
Figure 6: Ward Graph K=3



4.2.7 Manual Clustering of Points

I plotted the data set provided for this exercise. The graph on Fig 7 shows in highlighted green the way I would cluster the points with $k = 3$. This is the same result shown for the five agglomerative methods.

Figure 7: Manual Point Clustering with K=3

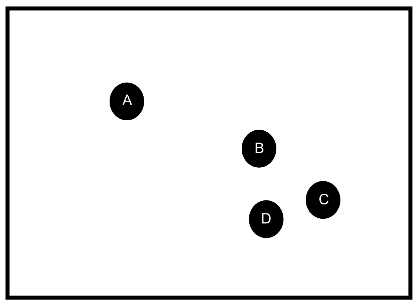


5 Exercise 9.10

Nearest neighbor clusters are not symmetric, in the sense that if instance A is one of instance Bs nearest neighbors, the reverse is not necessarily true. Explain how this can happen with a diagram.

Figure 8 is an instance which shows nearest neighbor clusters are not symmetric. For example, the figure shows that if we were going to cluster the set with $k = 2$, the nearest neighbor for A would be B, however for B the nearest neighbors are D and C. Proving that nearest neighbor clusters are not symmetric.

Figure 8: Asymmetric Nature of Nearest Neighbor



References

[1] T. S. W.B. Croft, D. Metzler, *Search Engine Information Retrieval in Practice*, Pearson Education, 2015.