Aalto University
School of Science

CS-C3160 - Data Science

Jaakko Hollmen, Department of Computer Science

29.10.2018-17.12.2018

Contents of the lecture

- In the two last chapters we look at *discrete patterns* instead of real valued variables
- Discrete methods for analyzing large binary datasets
- Frequent itemsets and breadth-first search

Large binary datasets

- Many rows (variables), many columns (observations)
- 0/1 data: does that thing occur in this observation?
- The cell $D(m, h)$ of matrix (table) D tells us whether the phenomenon described by variable m is present in observation h: $D(m, h) = 1$ or $D(m, h) = 0$.

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & \dots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & 0 & 0 & 0 & \dots & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & \dots & 1 & 0 \end{bmatrix}$$

Binary data example: Words appearing in documents

- Observation: document
- Variable: word (in its basic form)
- Data $D(s, d) = 1$: did the word s appear at least once in document d?

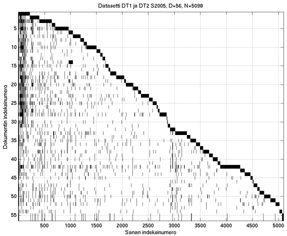
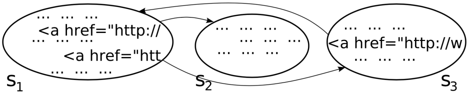


Figure: Word-document matrix from the computer exercises (Round 5). Black stripe = word s appears in document d; $D(s, d) = 1$. A corpus of 56 documents with 5098 different words, including two intentionally generated "copies".

Binary data example: Web pages

- Observation: Web page
- Variable: Web page
- $D(s, p) = 1$ if and only if page s links to page p

(s, p)	s_1	s_2	s_3
s_1	0	1	1
s_2	0	0	0
s_3	1	0	0



Binary data example: Mammal populations in nature

- Observation: 50km $\times$ 50km square *r*
- Variable: Mammal species *l*
- $D(l, r) = 1$ if and only if species *l* was observed in square *r*

Binary data example: Molecules within larger molecules

- Observation: molecule *m* (typically large)
- Observation: molecule *p* (smaller)
- $D(p, m) = 1$ if and only if molecule *p* exists as part of molecule *m*

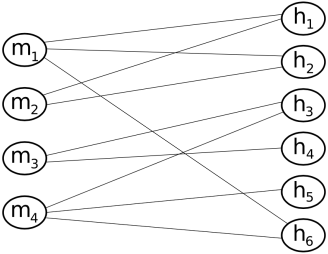
NSF 0-1 document example: Basic statistics

- 128000 documents, 30800 words – not a very large set of documents
- The table has $30800 \times 12800 = 3942400000 \approx 4 \cdot 10^9$ cells
- There are 10449902 (0.265%) ones in the dataset; no point in representing zeros
- There are approximately 81 different words in each document (1 = the word appears at least once)
- Each word appears in about 340 documents
- The distribution is skewed: some words appear often, others very rarely
- Some very common words have been omitted (“blacklist”)

Binary dataset as a network

- Network (or graph): a set of nodes (partially) connected with edges
- A binary dataset is easy to turn into a bipartite network: variables and observations are nodes, with an edge connecting them if and only if the corresponding data value is 1

	h_1	h_2	h_3	h_4	h_5	h_6
m_1	1	0	1	0	0	1
m_2	1	1	0	0	0	0
m_3	0	0	1	1	0	0
m_4	0	0	1	0	1	1



What could we look for in data like this?

- What does the data contain?
- What kind of groups do the variables and observations form?
- How do the variables depend on each other?
- Etc.
- Analysis of variable and observation degree (how many outbound edges does this variable have = how many observations involve this variable): power laws - a very active area of research
- *How do we look for small interesting sets of variables?*

Searching for commonly appearing combinations of variables: Notation

$$\boldsymbol{x} = \begin{bmatrix} x_{11} & x_{12} & x_1 3 & \cdot & x_{1n} \\ x_{21} & x_{22} & & & \cdot \\ x_{31} & & \cdot & & \cdot \\ \cdot & & & \cdot & \cdot \\ \cdot & & & \cdot & \cdot \\ x_{d1} & & & & x_{dn} \end{bmatrix}$$

- d-dimensional observation vector $\boldsymbol{x}$; its cells are $x_1, x_2, \ldots, x_d$
- Many observation vectors $\boldsymbol{x}(1), \ldots, \boldsymbol{x}(n)$
- Value of variable i in observation vector $\boldsymbol{x}(j)$ is denoted with $x(i, j)$ or x_{ij}

Frequent itemsets: Definition

- If there are eg. 30000 variables, it's not possible to search for all pairwise correlations
- How about variables commonly appearing together?
- Search for all variable pairs (a, b) so that there are at least N observations $\boldsymbol{x}(i)$ where $x(a, i) = 1$ and $x(b, i) = 1$
- *Generally:* Search for all variable sets $\{a_1, a_2, \ldots, a_k\}$ so that there are at least N observations $\boldsymbol{x}(i)$ where $x(a_1, i) = 1$ and $x(a_1, i) = 1$ and ... and $x(a_k, i) = 1$
- Let's call this variable set $\{a_1, a_2, \ldots, a_k\}$ a *frequent itemset*

Frequent itemsets: Example

- We have 4 shopping bags, each of which either contain or don't contain oranges (a), bananas (b), and apples (c).
- Bag #1: 5 oranges, 2 bananas. Bag #2: 4 bananas. Bag #3: 1 orange, 2 bananas, 1 apple. Bag #4: 8 oranges, 2 apples.
- 1 = bag contains item, 0 = bag does not contain item:

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

- With threshold value $N = 2$, frequent itemsets are $\{a\}, \{b\}, \{c\}, \{a, b\}$ and $\{a, c\}$. For example, oranges (a) and apples (c) appear together ($\{a, c\}$) in at least two bags.

How many frequent itemsets are there?

- If the data only contains ones, all pairs and subsets of variables fulfill the criterion
- In order for the problem to be sensible, the data has to be *sparse*. This is typically the case. In the previous example both shopping bags and products number in the thousands, but a customer only buys a small number of products at one time. The sparse matrix has way more zeros than ones.
- Using machine learning terminology: we look for commonly occurring positive conjunctions $(A \cap B)$

How do we search for frequent itemsets?

- Trivial approach: brute force search through all subsets of variables
- d variables $\Rightarrow 2^d$ subsets of variables
- If there are e.g. 100 variables, there are way too many subsets
- We have to be more clever
- *Basic observation:* if a set of variables $\{a_1, a_2, \ldots, a_k\}$ is frequent, all its subsets are frequent
- *Conversely:* A set of variables $\{a_1, a_2, \ldots, a_k\}$ can be frequent only if all of its subsets are frequent. The frequentness of subsets is a necessary but not sufficient condition.

Example of frequent itemset search 1/3

- Let's assume the variable sets $\{a\}, \{b\}, \{c\}, \{e\}, \{f\}, \{g\}$ (set size 1) occur often enough (we have at least N observations of each)
- Then the pair (i, j) is a *candidate* frequent itemset, when $i, j \in \{a, b, c, e, f, g\}$. There are $\binom{6}{2} = \frac{6!}{4! \cdot 2!} = 15$ candidate pairs.
- Let's assume that only the pairs $\{a, b\}, \{a, c\}, \{a, e\}, \{a, f\}, \{b, c\}, \{b, e\}, \{c, g\}$ are frequent (size 2) i.e. they occur together at least N times
- Let's combine these pairs into sets of size 3. The next step is to remove those sets which include non-frequent sets of size 2. Now we have candidate frequent itemsets of size 3.
  - For instance frequent pairs $\{a, b\}$ and $\{a, f\}$ can be combined as $\{a, b, f\}$, but then the subset $\{b, f\}$ is not frequent, so $\{a, b, f\}$ is not accepted as a candidate

Example of frequent itemset search 2/3

- Then $\{a, b, c\}$ and $\{a, b, e\}$ are the only possible frequent itemsets of size 3, because *all their subsets* are frequent
- Let's assume that $\{a, b, c\}$ is a frequent itemset (of size 3)
- We also already know that there can't be a frequent itemset of size 4 (or larger), because the subsets of $\{a, b, c, e\}$ include e.g. $\{a, c, e\}$ which is not frequent. In other words: the non-frequent set $\{a, c, e\}$ can't be turned into a frequent itemset by adding variables.
- As we now know that candidates of size 4 can't exist, the search ends.

Example of frequent itemset search 3/3

- The final step is to list all frequent itemsets $\{\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3\}$: $\{\{\}, \{a\}, \{b\}, \{c\}, \{e\}, \{f\}, \{g\}, \{a, b\}, \{a, c\}, \{a, e\}, \{a, f\}, \{b, c\}, \{b, e\}, \{c, g\}, \{a, b, c\}\}$
- Here "assume" means "has been computed from data". In this example we did not have the actual data (matrix $\boldsymbol{X}$).

Breadth-first search: Principle

- *Breadth-first search* (aka levelwise algorithm, a priori algorithm) is a simple but effective algorithm for frequent itemset search
- First, we look for frequent itemsets of size 1, then size 2, etc.
- The basic observation we made earlier becomes very useful: a set can be frequent only if all its subsets are frequent

Breadth-first search: Frequent itemset of size 1

- Frequent itemsets of size 1: variable *a* that has at least *N* ones in the dataset i.e. there are at least *N* observations (columns) *i* for which $x(a, i) = 1$. These are easy to find by going through the data once and summing the occurrences of each variable.

Breadth-first search: Frequent itemset of size 2

- When we know the frequent itemsets of size 1, we form *candidate* sets of size 2
- The set $\{a, b\}$ is a candidate set when both $\{a\}$ and $\{b\}$ are frequent
- Let's go through the data again, noting for each candidate set how many observations have a 1 for both variables *a* and *b*.
- Now we have found the frequent itemsets of size 2.

Breadth-first search: Frequent itemset of size k 1/2

- Let's assume that we know all the frequent itemsets of size k; let's call this collection $\mathcal{C}_k$
- Let's form *candidate sets* of size $k+1$: sets that could be frequent
  - A candidate set is a set of $k+1$ cells where all its subsets are frequent. It is enough to check that all subsets of size k are frequent.
  - Let's take all pairs of sets $A = \{a_1, \ldots, a_k\}$ and $B = \{b_1, \ldots, b_k\}$ from collection $\mathcal{C}_k$, compute their union $A \cup B$, check that the union has $k+1$ cells and that all subsets of size k are frequent (i.e. are in the collection $\mathcal{C}_k$).

Breadth-first search: Frequent itemset of size k 2/2

- When we have found the candidate sets of size $k+1$, we go through the data and note for each candidate set the number of columns where all the candidate set variables have the value 1
- Now we have found the frequent itemsets of size $k+1$
- The algorithm is repeated until no new candidate sets can be found

Frequent itemset search with threshold N = 10000

- Using threshold value $N = 10000$ we can find a total of 250 frequent itemsets in the 30800 variables (words) of the corpus, originating from 128000 documents
- At least 10000 documents have 9 different three-word sets

size	candidates	frequent itemsets
1	30800	134
2	8911	107
3	79	9
4	0	0

NSF example: Frequent itemset search with threshold N = 2000

- Using a smaller threshold value of $N = 2000$ we can find 16040 frequent itemsets and a much greater number of candidate sets
- There is one six-word set appearing in at least 2000 documents

size	candidates	frequent itemsets
1	30800	1171
2	685035	7862
3	105146	6098
4	5813	889
5	92	19
6	1	1

NSF example: Number of frequent itemsets as a function of threshold N

– Threshold *N*: there have to be at least *N* columns with the value 1

threshold N	frequent itemsets
10000	250
5000	1539
2000	16040
1000	96223
⋮	⋮

NSF example: Frequent itemset of size 6 with threshold N = 2000

```
21582 program
21614 project
23313 research
24416 science
26454 students
29137 university
```

How hard is it to look for frequent itemsets?

- We have 0/1 data and a threshold *N*, the task is to look for all frequent itemsets
- The answer can be very large
- In practice the a priori algorithm is efficient enough

Searching for commonly occurring patterns of different kind

- The same principles can be applied to other tasks
- Search for commonly occurring episodes
- Search for commonly occurring subnets
- Search for commonly occurring subtrees
- The same basic idea of the a priori algorithm works

Summary

In this chapter we studied how to transform discrete datasets of various forms into binary (0/1) data. Binary data can be processed and analyzed efficiently with e.g. the a priori algorithm.