

|  |  |
| --- | --- |
| **Name: K. Pavan karthik** | **Reg no: 17MIS1038** |
| **Course: Data Mining** | **Semester : Winter 19** |
| **Assignment:3** | **Slot:A1+TA1** |
| **Date: 17/02/2019** | **Prof: Dr.Nagaraju** |

1. Give a numerical example to demonstrate how we can detect univariate outliers based on the normal distribution. What are the difficulties in detecting outliers using histograms?

Ans) The data set of N = 90 ordered observations as shown below is examined for outliers:

30, 171, 184, 201, 212, 250, 265, 270, 272, 289, 305, 306, 322, 322, 336, 346, 351, 370, 390, 404, 409, 411, 436, 437, 439, 441, 444, 448, 451, 453, 470, 480, 482, 487, 494, 495, 499, 503, 514, 521, 522, 527, 548, 550, 559, 560, 570, 572, 574, 578, 585, 592, 592, 607, 616, 618, 621, 629, 637, 638, 640, 656, 668, 707, 709, 719, 737, 739, 752, 758, 766, 792, 792, 794, 802, 818, 830, 832, 843, 858, 860, 869, 918, 925, 953, 991, 1000, 1005, 1068, 1441

The above data is available as a text file.

The computations are as follows:

Median = (n+1)/2 largest data point = the average of the 45th and 46th ordered points = (559 + 560)/2 = 559.5

Lower quartile = .25(N+1)th ordered point = 22.75th ordered point = 411 + .75(436-411) = 429.75

Upper quartile = .75(N+1)th ordered point = 68.25th ordered point = 739 +.25(752-739) = 742.25

Interquartile range = 742.25 - 429.75 = 312.5

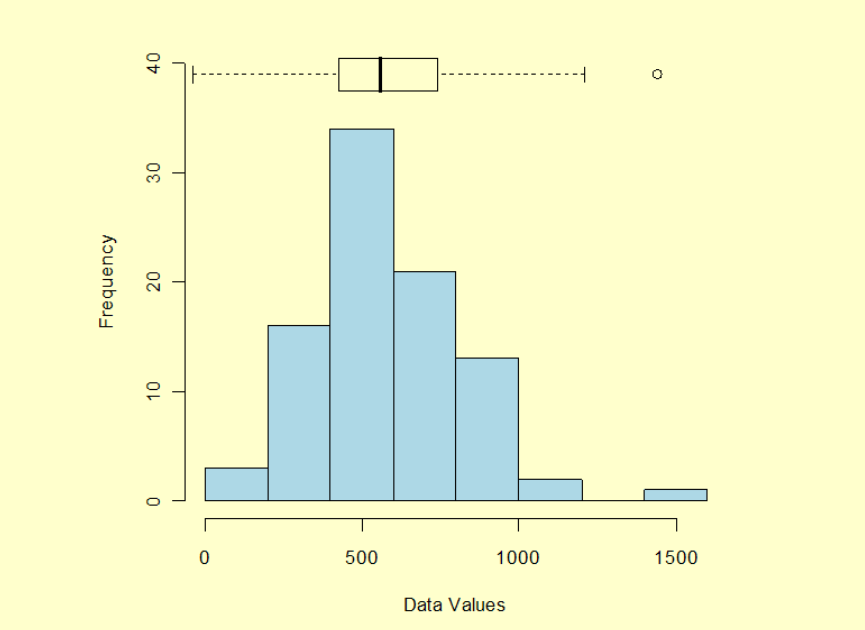
Lower inner fence = 429.75 - 1.5 (312.5) = -39.0

Upper inner fence = 742.25 + 1.5 (312.5) = 1211.0

Lower outer fence = 429.75 - 3.0 (312.5) = -507.75

Upper outer fence = 742.25 + 3.0 (312.5) = 1679.75

From an examination of the fence points and the data, one point (1441) exceeds the upper inner fence and stands out as a mild outlier; there are no extreme outliers.



The outlier is identified as the largest value in the data set, 1441, and appears as the circle to the right of the box plot.

Outliers should be investigated carefully. Often they contain valuable information about the process under investigation or the data gathering and recording process. Before considering the possible elimination of these points from the data, one should try to understand why they appeared and whether it is likely similar values will continue to appear. Of course, outliers are often bad data points.

1. Describe some algorithms used in Web mining. Describe some algorithms for calculating sequential patterns, forward sequences and maximal frequent sequences

ANs) PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages. According to Google:

It is not the only algorithm used by Google to order search engine results, but it is the first algorithm that was used by the company, and it is the best-known.

The above centrality measure is not implemented for multi-graphs.

The PageRank algorithm outputs a probability distribution used to represent the likelihood that a person randomly clicking on links will arrive at any particular page. PageRank can be calculated for collections of documents of any size. It is assumed in several research papers that the distribution is evenly divided among all documents in the collection at the beginning of the computational process. The PageRank computations require several passes, called “iterations”, through the collection to adjust approximate PageRank values to more closely reflect the theoretical true value.

In the HITS algorithm, the first step is to retrieve the most relevant pages to the search query. This set is called the *root set* and can be obtained by taking the top pages returned by a text-based search algorithm. A *base set* is generated by augmenting the root set with all the web pages that are linked from it and some of the pages that link to it. The web pages in the base set and all hyperlinks among those pages form a focused subgraph. The HITS computation is performed only on this *focused subgraph*. According to Kleinberg the reason for constructing a base set is to ensure that most (or many) of the strongest authorities are included.

Authority and hub values are defined in terms of one another in a [mutual recursion](https://en.wikipedia.org/wiki/Mutual_recursion). An authority value is computed as the sum of the scaled hub values that point to that page. A hub value is the sum of the scaled authority values of the pages it points to. Some implementations also consider the relevance of the linked pages.

The algorithm performs a series of iterations, each consisting of two basic steps:

* **Authority update**: Update each node's *authority score* to be equal to the sum of the *hub scores* of each node that points to it. That is, a node is given a high authority score by being linked from pages that are recognized as Hubs for information.
* **Hub update**: Update each node's *hub score* to be equal to the sum of the *authority scores* of each node that it points to. That is, a node is given a high hub score by linking to nodes that are considered to be authorities on the subject.

# Clever:

Identify authoritative and hub pages.

Authoritative Pages :

Highly important pages.

Best source for requested information.

Hub Pages :

Contain links to highly important pages.

Hits Algorithm(Overcome Constraints in the Clever)

Hyperlink Induced Topic Search (HITS) algorithm ranks the web page by processing in links and out links of the web pages. Kleinberg identifies two different forms of Web pages called hubs and authorities. Authorities are pages having important contents. Hubs are pages that act as resource lists, guiding users to authorities. Thus, a good hub page for a subject points to many authoritative pages on that content, and a good authority page is pointed by many good hub pages on the same subject. Hubs and Authorities are shown in Figure 4. Kleinberg says that a page may be a good hub and a good authority at the same time. This circular relationship leads to the definition of an iterative algorithm called HITS.

The Hub score and Authority score for a node is calculated with the following algorithm:

Start with each node having a hub score and authority score of 1.

Run the authority update rule

Run the hub update rule

Normalize the values by dividing each Hub score by square root of the sum of the squares of all Hub scores, and dividing each Authority score by square root of the sum of the squares of all Authority scores.

Repeat from the second step as necessary.

HITS, like Page and Brin's PageRank, is an iterative algorithm based on the linkage of the documents on the web. However it does have some major differences:

It is query dependent, that is, the (Hubs and Authority) scores resulting from the link analysis are influenced by the search terms;

As a corollary, it is executed at query time, not at indexing time, with the associated hit on performance that accompanies query-time processing.

It computes two scores per document, hub and authority, as opposed to a single score;

It is processed on a small subset of ‘relevant’ documents (a 'focused subgraph' or base set), not all documents as was the case with PageRank.

B)

Commonly used algorithms include:

GSP algorithm

Sequential PAttern Discovery using Equivalence classes (SPADE)

FreeSpan

PrefixSpan

MAPres

GSP algorithm (Generalized Sequential Pattern algorithm) is an algorithm used for sequence mining. The algorithms for solving sequence mining problems are mostly based on the a priori (level-wise) algorithm. One way to use the level-wise paradigm is to first discover all the frequent items in a level-wise fashion. It simply means counting the occurrences of all singleton elements in the database. Then, the transactions are filtered by removing the non-frequent items. At the end of this step, each transaction consists of only the frequent elements it originally contained. This modified database becomes an input to the GSP algorithm. This process requires one pass over the whole database.

GSP algorithm makes multiple database passes. In the first pass, all single items (1-sequences) are counted. From the frequent items, a set of candidate 2-sequences are formed, and another pass is made to identify their frequency. The frequent 2-sequences are used to generate the candidate 3-sequences, and this process is repeated until no more frequent sequences are found. There are two main steps in the algorithm.

# Sequential PAttern Discovery using Equivalence classes (SPADE)

Sequential Pattern Discovery using Equivalence classes

Identifies patterns by traversing lattice in a top down manner.

Divides lattice into equivalent classes and searches each separately.

ID-List: Associates customers and transactions with each item.

Sequential pattern mining is a topic of data mining concerned with finding statistically relevant patterns between data examples where the values are delivered in a sequence.[1] It is usually presumed that the values are discrete, and thus time series mining is closely related, but usually considered a different activity. Sequential pattern mining is a special case of structured data mining.

There are several key traditional computational problems addressed within this field. These include building efficient databases and indexes for sequence information, extracting the frequently occurring patterns, comparing sequences for similarity, and recovering missing sequence members. In general, sequence mining problems can be classified as string mining which is typically based on string processing algorithms and itemset mining which is typically based on association rule learning. Local process models [2] extend sequential pattern mining to more complex patterns that can include (exclusive) choices, loops, and concurrency constructs in addition to the sequential ordering construct.

# Algorithm For Forward Sequences:

1. Computing Forward Probabilities Then the forward probability for state i at time t, at (i), is the probability that an HMM will output a sequence O1,t and end in state si . For instance, the forward probability of being in state s1 after two steps on the sequence R W B B is the joint probability
2. Backward Probability The other probability we need is called the backward probability, which is the probability of starting in state Si at time t and generating the rest of the observation sequence ot+1,..., oT. bt(i) = Prob(ot+1,..., oT | Qt = Si ) The backward probabilities can be computed efficiently using an algorithm that is a simple “backwards” variant of the forward algorithm. Rather than starting at time 1, the algorithm starts at time T and then works backwards through the network from observation oT down to ot+1. The initial probabilities of being in state Si at time T and generating nothing else is simply 1 since there is no more output in this context.
3. Using Forward and Backwards Probabilities With both the forward and backward probabilities defined, we can now define the probability of observing o1 ... oT and being in state Si at time t as follows: Prob(o1 ... oT, qt = Si) = at(i) \* bt(i) i.e., its the probability of observing o1 ... ot ending in state Si, times the probability of observing ot+1 ... oT given that we start in Si. With this, we can now compute the conditional probability of being in state Si at time t given the observation sequence

# Algorithms for Maximal Frequent Sequences:

The most important purpose of the document or text database is to provide an efficient and flexible query system for the non-technical users. The query operation usually includes some kind of filtering or selection operation. The selection criteria are usually given by a list of keywords and a list of matching documents is returned. In the naive query evaluation method every document is tested in a sequential way for matching. The documents containing the given keywords are inserted into the result list. The main drawback of this evaluation method is the low efficiency. Due to the large number of documents this processing is very costly and takes a long time. The purpose of the researches is to improve the efficiency of the query processing algorithms in text databases. One of the most useful methods of cost reduction is the pre-clustering or preclassification of the documents. Based on the created cluster index structure, the set of tested documents can be reduced drastically. Another benefit of this method is that the document cluster hierarchy can be applied directly for query processing. The user may navigate in the hierarchy performing an interactive query based on relevance feedback. Many proposals have been published in the literature related to this topic, but due to the complexity of the semantic based evaluation of the documents, the problem of efficient text clustering can not be regarded as a closed, finished research area. There are may open questions to be solved in the near future. This paper is devoted to present an efficient algorithm for determining the frequent word sequences in documents. First, an overview is given about the main clustering and classification methods, then the importance of word sequences is described. In the next, an algorithm is given for determining the frequent word sequences.