Aalto University
School of Science
Degree Programme in Computer Science and Engineering

Stella Student

Software Processes for Dummies: Re-inventing the Wheel

Master's Thesis
Espoo, June 18, 2011

DRAFT! — August 23, 2016 — DRAFT!

Supervisors:	Professor Antti Ylä-Jääski, Aalto University
	Professor Pekka Perustieteilijä, University of Helsinki
Advisor:	Olli Ohjaaja M.Sc. (Tech.)

Aalto University
School of Science
Degree Programme in Computer Science and Engineering

ABSTRACT OF
MASTER'S THESIS

Author:	Stella Student		
Title: Software Processes for Dummies: Re-inventing the Wheel			
Date:	June 18, 2011	**Pages:**	48
Major:	Data Communication Software	**Code:**	T-110
Supervisors:	Professor Antti Ylä-Jääski Professor Pekka Perustieteilijä		
Advisor:	Olli Ohjaaja M.Sc. (Tech.)		

A dissertation or thesis is a document submitted in support of candidature for a degree or professional qualification presenting the author's research and findings. In some countries/universities, the word thesis or a cognate is used as part of a bachelor's or master's course, while dissertation is normally applied to a doctorate, whilst, in others, the reverse is true.

!FIXME **Abstract text goes here (and this is an example how to use fixme).** FIXME! Fixme is a command that helps you identify parts of your thesis that still require some work. When compiled in the custom `mydraft` mode, text parts tagged with fixmes are shown in bold and with fixme tags around them. When compiled in normal mode, the fixme-tagged text is shown normally (without special formatting). The draft mode also causes the "Draft" text to appear on the front page, alongside with the document compilation date. The custom `mydraft` mode is selected by the `mydraft` option given for the package `aalto-thesis`, near the top of the `thesis-example.tex` file.

The thesis example file (`thesis-example.tex`), all the chapter content files (`1introduction.tex` and so on), and the Aalto style file (`aalto-thesis.sty`) are commented with explanations on how the Aalto thesis works. The files also contain some examples on how to customize various details of the thesis layout, and of course the example text works as an example in itself. Please read the comments and the example text; that should get you well on your way!

Keywords: ocean, sea, marine, ocean mammal, marine mammal, whales, cetaceans, dolphins, porpoises

Language: English

Acknowledgements

I wish to thank all students who use LaTeX for formatting their theses, because theses formatted with LaTeX are just so nice.

Thank you, and keep up the good work!

Espoo, June 18, 2011

Stella Student

Abbreviations and Acronyms

2k/4k/8k mode	COFDM operation modes
3GPP	3rd Generation Partnership Project
ESP	Encapsulating Security Payload; An IPsec security protocol
FLUTE	The File Delivery over Unidirectional Transport protocol
e.g.	for example (do not list here this kind of common acronymbs or abbreviations, but only those that are essential for understanding the content of your thesis.
note	Note also, that this list is not compulsory, and should be omitted if you have only few abbreviations

Contents

Abbreviations and Acronyms **4**

1 Introduction **7**

1.1 Problem statement 8

1.2 Helpful hints 8

1.3 Structure of the Thesis 8

2 Background **9**

2.1 Language and Structure 9

2.2 Finding and referring to sources 10

2.2.1 Finding sources 10

2.2.2 Referring to sources 11

3 Clustering Methods **14**

3.1 K-Means 14

3.1.1 EM Algorithm in K-Means 15

3.1.2 Issues in K-Means 16

3.2 Density Based Clustering Methods 19

3.2.1 DBSCAN and DFS 19

3.2.2 Notions of Density in Clusters 21

3.2.3 Analysis of DBSCAN 23

4 Generative Methods **26**

4.1 Discrete Markov Process 26

4.2 Hidden Markov Models 29

4.2.1 Learning and Inference 31

5 Experiment **36**

5.1 Data Details and Representation 36

5.2 Methods 38

5.2.1 Choice of Clustering Method 39

5.2.2 Choice of Generative Method 40

6 Evaluation **41**

7 Discussion **42**

8 Conclusions **43**

A First appendix **47**

Chapter 1

Introduction

This is my master's thesis, and I am very proud of it. Of course, when I write my *real* master's thesis, I will not use the singular pronoun *I*, but rather try to avoid referring to myself and speak of the research *we* have conducted—I rarely work alone, after all. Yet, both *I* and *we* are correct, and it depends on the instructor and the supervisor (of course from you, too), which one they would prefer. Anyway, the tense should be active, and passive sentenses should be avoided (especially, writing sentences where the subject is presented with by preposition), so often you cannot avoid choosing between the pronouns. Life is strange, but there you have it.

By the way, the preferred order of writing your master's thesis is about the same as the outline of the thesis: you first discover your problem and write about that, then you find out what methods you should use and write about that. Then you do your implementation, and document that, and so on. However, the abstract and introduction are often easiest to write last. This is because these really cover the entire thesis, and there is no way you could know what to put in your abstract before you have actually done your implementation and evaluation. Rarely anyone write the thesis from the beginning to the end just one time, but the writing is more like process, where every piece of text is written at least twice. Be also prepared to delete your own text. In the first phase, you can hide it into comments that are started with % but during the writing, the many comments should be visible for your helpers, the instructor and supervisor.

The introduction in itself is rarely very long; two to five pages often suffice.

1.1 Problem statement

Undergraduate students studying technical subjects do not consider typography very interesting these days, and therefore the typographical quality of many theses is unacceptably low. We plan to rectify this situation somewhat by providing a decent-quality example thesis outline for students. We expect that the typographical quality of the master's theses will dramatically increase as the new thesis outline is taken into use.

1.2 Helpful hints

Read the information from the university master's thesis pages [4] before starting the thesis. You should also go through the thesis grading instructions [1] together with your instructor and/or supervisor in the beginning of your work.

1.3 Structure of the Thesis

You should use transition in your text, meaning that you should help the reader follow the thesis outline. Here, you tell what will be in each chapter of your thesis.

Chapter 2

Background

The problem must have some background, otherwise it is not interesting. You can explain the background here. Probably you should change the title to something that describes more the content of this chapter. Background consists of information that help other masters of the same degree program to understand the rest of the thesis.

Transitions mentioned in Section 1.3 are used also in the chapters and sections. For example, next in this chapter we tell how to use English language, how to find and refer to sources, and enlight different ways to include graphics in the thesis.

2.1 Language and Structure

Moreover, the transitions are also used in the paragraph and the sentence level meaning that all the text is linked together. For example, the word "moreover" here is one way, but of course you should use variation in the text. Examples of transitional devices (words) and their use can be found from writing guides, e.g. from the Online Writing Lab (OWL)[1] of Purdue University or Strunk's Elements of Style[2]. Remember that footnotes are additional information, and they are seldom used. If you refer to a source, you do no not use footnote. The right command for the references is *cite*.

The language used in the thesis should be technical (or scientifical). For example, the abreviations aren't used but all them are written open (i.e. "are not"). Since the content itself is often hard to understand (and explain), the sentences should not be very long, use complex language with several examples embedded in the same sentence, and, also, seldom used words and

[1]http://owl.english.purdue.edu/owl/resource/574/02/
[2]http://www.bartleby.com/141/

weird euphemism or paraphrases can make the sentence hard to follow and to read it with only one time, and making everything even harder to understand all this without any punctuation marks makes the instructor cry and finally after trying to correct the language, you will get boomerang, and everyone's time has just been wasted.

Please use proofreaders before sending even your unfinished version to the instructor and/or supervisor. You will get better comments when they do not need first proofread your text. Moreover, they can consentrate to the content better if the language and spelling mistakes are not distracting the reading. Several editors have their own proofreading tools, e.g. ispell in emacs. You can also use Microsoft Word to proofread your thesis: it can correct also some grammatical errors and not just misspelled words.

Note also that if you have a section or a subsection, you have to have at least two of them, or otherwise the section or subsection title is unnecessary. Same with the paragraphs an: you should not have sections with only one paragraphe, and single sentence paragraph. Furthermore, always write some text after the title before the next level title.

2.2 Finding and referring to sources

Never ever copy anything into your theses from somebody else's text (nor your own previously published text). Never. Not even for starting point to be rewritten later. The risk is that you forgot the copied text to your thesis and end up to be accused of plagiarism. Plagiarism is a serious crime in studies and science and can ruin your career even its beginning. To repeate: never cut and paste text into your thesis!

2.2.1 Finding sources

All work is based on someone else's work. You should find the relevant sources of your field and choose the best of them. Also, you should refer to the original source where a fact has been mentioned first time. Remember source evaluation (criticism) with all sources you find.

Good starting points for finding references in computer science are:

- Nelli Portal (Aalto Library): `http://www.nelliportaali.fi`
- ACM Digital library: `http://portal.acm.org/`
- IEEExplore: `http://ieeexplore.ieee.org`
- ScienceDirect: `http://www.sciencedirect.com/`

- . . . although Google Scholar (`http://scholar.google.com/`) will find links to most of the articles from the abovementioned sources, if you search from within the university network

Some of the publishers do not offer all the text of the articles freely, but the library has agreed on the rights to use the whole text. Thus, you should sometimes use computers in the domain of the university in order to get the full text. Sometimes the Nelli Portal can also help getting the whole article instead of just the abstract. The library has also brief instrucions how to find information [2].

Instead of normal Google, use Google Scholar (`http://scholar.google.fi/`). It finds academic publications whereas normal Google find too much commercial advertisements or otherwise biased information. Wikipedia articles should be referred to in the master thesis only very, very seldomly. You can use Wikipedia for understanding some basics and finding more sources, but often you cannot be sure if the article is correct and unbiased.

One important part of the sources that you have found is the reference list. This way you can find the original sources that all the other research of the field refer. Often you can also find more information with the name of the researchers that are often referred in the articles.

2.2.2 Referring to sources

The main point in referring to sources is to separate your own thinking and text from that of others. Facts of the research area can be given without reference, but otherwise you should refer to sources. This means two things: marking the source in the text where it has been used, and listing the sources usually in the end of the thesis in a way that help the reader to find the original source.

There are several bibliography styles, meaning how to form the bibliography in the end of the thesis. Aalto's library has good instructions for many styles [3]. You should ask from your supervisor or instructors which style you should use. This thesis template uses the number style that is often used in software engineering. The other style also used in the CS field, e.g. usability, is the Harvard style where instead of numbers, the reference is marked into the text with author's name and publishing year. Other areas use also many other styles for making the lists and marking the references.

In addition to the list in the end of the thesis, you have to mark the source in the text where the source is used. There are three places for the reference: in a sentence before the period, in the end of a sentence after the period, or in the end of a paragraph. All of them have different meaning.

The main point is that first you paraphrase the source using your own words and then mark the source. Next, we give short examples that are marked with *emphasised text.*

Haapasalo [13] researched database algorithms that allows use of previous versions of the content stored in the database. This kind of marking means that this paragraph (or until the next reference is given) is based on the source mentioned in the beginning. Giving the source you should use only the family name of the first author of the article, and not give any hints about what is the type of the article that is referred.

B+-trees offers one way to index data that is stored in to a database. Multiversion B+-trees (MVBT) offer also a way to restore the data from previous versions of the database. Concurrent MVBT allows many simultaneous updates to the database that is was not possible with MVBT. [13] When the marking is after the period, the reference is retrospective: all the paragraph (or after previous reference marking) is based on the source given in its end. If the content is very broad, you can start with saying *According to Haapasalo*, then continue referring the source with several separate sentences, and in the end put the marking of your source *that shows that CMVBT are the best. [13].*

If your paragraph has several sources, the above mentioned styles are not proper. The reader of your thesis cannot know which of your sources give which of the statements. In this case, it is better to use more finegraded refering where the reference markings that are embedded in the sentences. For example, *the multiversion B+-tree (MVBT) index of Becker et al. [6] allows database users to query old versions of the database, but the index is not transactional. It's successor, the transactional MBVT (TMVBT), allows a single transaction running in its own thread or process to update the database concurrently with other transactions that only read the database [14]. Further development, titled the concurrent MBVT (CMVBT), allows several transactions to perform updates to the database at the same time [13].* Here, the references are marked before the period in the sentences where they are used.

Finally, direct quotes are allowed. However, often you should avoid them since they do not usually fit in to your text very well. Using direct quotes has two tricks: quotation marks and the source. *"Even though deletions in a multiversion index must not physically delete the history of the data items, queries and range scans can become more efficient, if the leaf pages of the index structure are merged to retain optimality." [13]* Quotes are hard to make neatly since you should use only as much as needed without changing the text. Moreover, you often do not really understand what the author has mentioned with his wordings if you cannot write the same with your own words. Remember also that never cut and paste anything without marking the quotation marks right away, and in general, never cut and paste anything at all!

Sometimes getting the original source can be almost impossible. In an extremely desperate situation, you can refer with structure *mr X [...] according to ms Y [...] defined that*, if you find a source that refers to the original source. Note also that the reference marking is never used as sentence element (example of how **not** to do it: *[13] describes an optimal algorithm for indexing multiversiond databases.*).

Chapter 3

Clustering Methods

As described in Chapter XX, all patients data is retrieved from the database directly without any manual labeling. Due to lack of labels, unsupervised learning algorithms should be adopted. Clustering is a collection of unsupervised methods, which identifies groups of data points according to a defined similarity metric, such that objects in the same group posses higher similarities compared to objects in other groups. The clustering process does not rely on labels but the choice of similarity metrics. Variations in similarity metrics lead to different clustering methods.

Applying clustering methods in anomaly detection tasks has been studied numerously [15] . This chapter introduces two typical methods, K-Means [18] and DBSCAN [11]. Problem formulation, solutions, and potential issues are formally described using elaborated notations in following sections. However, analysis on the performance and constraints of these two methods are postponed to Chapter XXX, which reveals their practicality and infeasibility in the previously described anomaly detection problem. (Time, Memory consumption. For K-Means, definition of center.)

3.1 K-Means

K-Means is one of the simplest unsupervised algorithm which solves clustering problem. Despite its simplicity, K-Means has gained success in various situations, including anomaly detection [15] [8], image segmentation and compression [12], and preprocessing for more complicated algorithms. The method can be formally defined as follow: Given a data set $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ consisting of N observations in D-dimensional space, the object is to partition the data into K groups, by defining a set of K centers $\{\boldsymbol{\mu}_1, ..., \boldsymbol{\mu}_k\}$ in the same space, and assigning each observation to exactly one center point. Each center point represents a prototype associated with the k^{th} cluster.

The assignments can be represented using 1-of-K scheme. Then, for each data point x_n, a corresponding K-dimensional variable consisting of K binary elements $r_{nk} \in \{0, 1\}$ is introduced. Among these r_{nk}, exactly one of them equals 1, which means $\mathbf{x}_n$ belongs to the k^{th} cluster. Using this notation, evaluation of the clustering quality can be defined using the object function as follow:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} D(\mathbf{x}_N - \boldsymbol{\mu}_k) \tag{3.1}$$

where D is the dissimilarity metric. Common choice of the metric is l_1-norm or l_2-norm. Mahalanobis distance is also adopted while considering the covariances between the K-dimensions [9]. In the following context, l_2-norm is adoped for discussion. Intuitively, this function can be considered as the distance summation of each point to its corresponding cluster prototype $\boldsymbol{\mu}_k$. The K-Means aims at finding a set of $\boldsymbol{\mu}_k$ which minimizes the object function. Finding the optimal solution for the above object function proves to be NP-Hard [5]. However, employing heuristic algorithms enables finding converged local optimal solutions. Section 3.1.1 describes one iterative algorithm, EM. Section 3.1.2 explores common issues related to K-Means and remedies.

3.1.1 EM Algorithm in K-Means

EM algorithm is an iterative algorithm to find local maximum. Each iteration consists of two phases, Expectation and Maximization, which corresponds to minimize the objective function J with respect to r_{nk} and $\boldsymbol{\mu}_k$ respectively. In the E(expectation) step, the algorithm minimize J with respect to r_{nk}, while keeping the $\boldsymbol{\mu}_k$ fixed. Then in the following M(maximization) step, the algorithm minimizes J with respect to $\boldsymbol{\mu}_k$, while keeping r_{nk} fixed.

Considering the optimization in E step, a critical observation of (3.1) is that $\mathbf{r}_n$'s are independent of each other. Thus, optimization on $\mathbf{r}_n$'s can be done seperately for each n. The solution is simply setting the r_{nk} corresponding to the minimum $\|\mathbf{x}_n - \boldsymbol{\mu}_k\|^2$ to 1. Intuitively, the algorithm assigns $\mathbf{x}_n$ to the closest cluster center. Formally, the solution can be written as

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \parallel \mathbf{x}_n - \mu_j \|^2 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

In the following M step, the above determined r_{nk} is clamped. Then, $\boldsymbol{\mu}_k$ appears only in a quadratic term in J. Setting derivatives of J with respect to $\boldsymbol{\mu}_k$ to zero, solution formula for $\boldsymbol{\mu}_k$ can be expressed in following closed form

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\mathbf{x}_n}{\sum_n r_{nk}} \tag{3.3}$$

This step can be considered as recomputing the cluster prototype by setting $\boldsymbol{\mu}_k$ to the mean of all points assigned to that cluster.

EM keeps executing these two steps alternatively, until a convergence happens or until number of iterations exceeded a predifined value. Since in each phase, one variable is fixed and updating another variable minimizes the cost function J, convergence is guaranteed. Formal proof on convergence has been studied by MacQueen [19].

The algorithm is illustrated using dataset generated independently from three gaussian distributions in Figure 3.1. In this demonstration, the algorithm takes $K = 3$, which is the correct number of components. Before running the first iteration, initialized $\boldsymbol{\mu}_k$ is required. This initialization is done by choosing three objects from the data set randomly.

3.1.2 Issues in K-Means

Despite the simplicity of K-Means, several underlying issues exists. The first potential is that, how to choose the value for K. In the above illustration, K was set to 3 which is the correct number of components. However, if K wasn't set to the correct value, unsatisfied clustering may be generated. Example of this issue is shown in Figure 3.2(a)-(c). To solve this problem, one practical way is drawing graph of the cost function versus value of K, as shown in Figure 3.2(d). Intuitively, when K is under smaller than the true number of clusters, increasing K will lead to a huge drop of cost function value. However, when K has reached or exceeded the correct value, incresing K leads only small cost function value drop. Thus, number of clusters corresponding to the 'elbow' point can be considered as the real number of clusters.

Another issues of K-Means relates to the initialization of $\boldsymbol{\mu}_k$. Since EM finds only local optimal solutions, a poor initialization could lead to worse clustering result. To avoid this problem, it is practical to run K-Means for several times and choose the best result according to the value of the cost function.

One more critical issue of K-Means lies in the choice of similarity metric. As mentioned at the beginning of this chapter, variations in similarity metric leads to different clustering methods. Choosing l-2 norm is convenient in terms of computation, but this choice limits the type of data variables to certain types. Using l-2 norm on categorical data is inappropriate since no ordering of categorical values exists. Also, this makes computing the mean

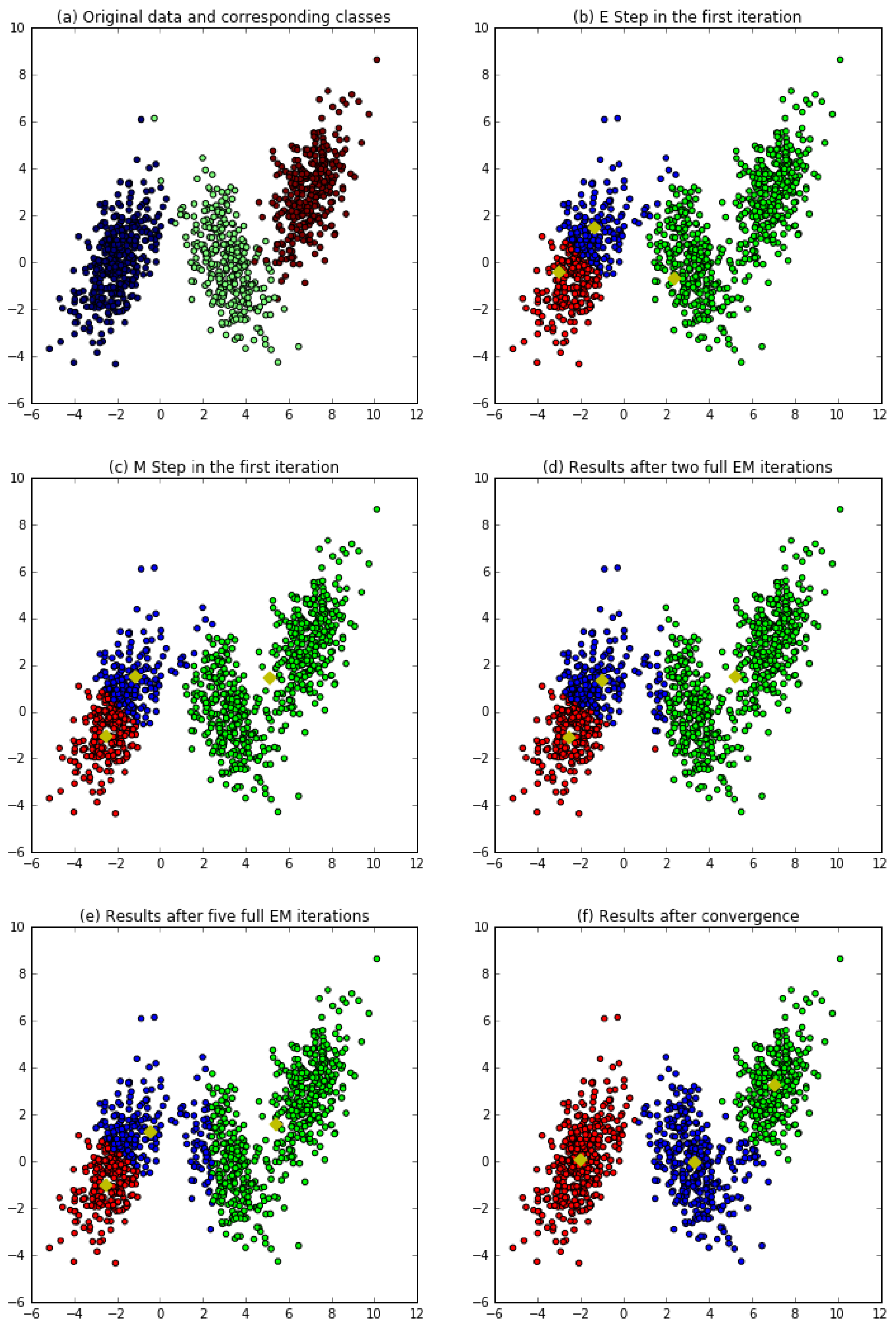

Figure 3.1: Illustraion of EM algorithm on K-Means using data generated independently from three gaussian distribution. (a) Original data and corresponding classes. Classes are denoted in different colors. (b) Assignments of each data after the first E step. The yellow diamonds represent the intialized $\boldsymbol{\mu}_k$. (c) Updated $\boldsymbol{\mu}_k$ after the M step in the first iteration. (d)-(f) Clustering results after several successive full EM iterations until convergence is met.

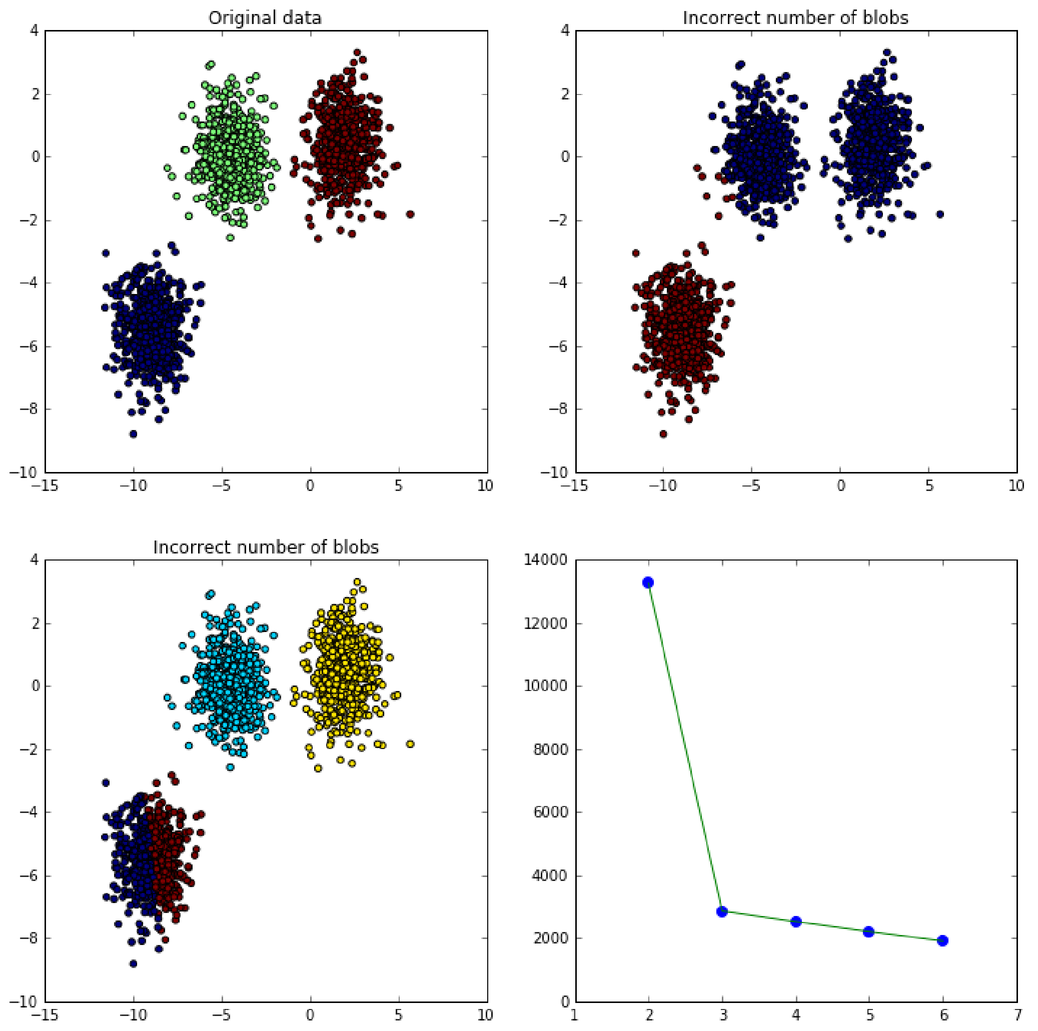

Figure 3.2: Issues of K-Means incurred by choosing inappropriate value for K

value a hard problem. To use K-Means on other data types, the similarity metric should be elaborately designed.

Besides, K-Means tends to form clusters into a convex space. As shown in Figure 3.1(b)-(e), the boundary between two different clusters forms a line lying at the midway and is perpendicular to the line connecting two cluster prototypes. However, a cluster is not necessary to be convex. This also limits the application of K-Means.

Finally, K-Means is also sensitive to noises. As shown in formula 3.3, updating $\boldsymbol{\mu}_k$ involves computing the mean of all data points assigned to that cluster. When noise objects with large deviation from other points in this cluster exist, update of $\boldsymbol{\mu}_k$ will be strongly affected.

3.2 Density Based Clustering Methods

This section explores another type of clustering method which forms cluster from the view of density aspect. **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (DBSCAN) [11] considers to be one of the most successful method from this category [17]. Compared to K-Means, DBSCAN posseses following advantages: 1). DBSCAN can detect clusters of arbitrary shape. 2). DBSCAN can determine the number of clusters automatically. 3). DBSCAN is robust to noises.

The following of this chapter is organized as follows. Section 3.2.1 compares and reveals the relation of DBSCAN to a basic graph traversal algorithm **D**epth **F**irst **S**earch(DFS). This section will introduce the critical concept used by DBSCAN which differentiate it with DFS. Section 3.2.2 describe this concept in more details. Section 3.2.3 compares DBSCAN and K-Means, and discusses practical issues of DBSCAN.

3.2.1 DBSCAN and DFS

Before introducing DBSCAN, it would be helpful to review a basic graph traversal algorithm, DFS, which is an algorithm which traverses or visits a graph. In the discussion of this context, the whole process is divided into two procedures DFS and DFS-Visit. A typical recursive implementation is described as follow.

```
DFS($G$)
1  for each vertex $u \in G$
2      $u.visited$ = FALSE
3  for each vertex $u \in G$
4      if $u.visited$ == FALSE
5          DFS-VISIT($G, u$)
```

```
DFS-VISIT($G, u$)
1  $u.visited$ = TRUE
2  for each vertex $v \in G.Adj[u]$
3      if $v.visited$ == FALSE
4          DFS-VISIT($G, v$)
```

The first two lines in DFS initializes the algorithm by setting all nodes to be unvisited. Then, the algorithm traverse vertices in the graph. Once an unvisited node is found, DFS-VISIT is called. In the procedure DFS-VISIT, the given node u is labled as visited. If an unvisited child v of u is found, this procedure goes one layer deeper by calling another DFS-VISIT on v. After finishing DFS-VISIT(G, u), the algorithm backtrack to visit other unvisited children of u, which are siblings of v.

Assume DFS executes on an undirected graph. Each call of DFS-VISIT on an unvisited node u explores u and all nodes reachable to it. These nodes form a component which disconnects with other components by other runs of DFS-VISIT. Thus, each component can be considered as a cluster. The criterion to form a cluster is that each pair of nodes can reache each other through a path consisting of nodes only in this cluster.

DBSCAN can be seen as an application of DFS. However, DBSCAN differs from standard DFS from its useage of DFS-VISIT. In standard DFS, the algorithm goes deeper by calling DFS-VISIT(G, v) on all unvisited children of node u. In DBSCAN, however, the algorithm goes deeper if and only if node u satisfies extra conditions, which are specified by two user given value Eps and $MinPts$. These extra conditions make the component generated from DBSCAN a meaningful cluster viewing from the density side. The pseudocode is shown below.

```
DBSCAN($S, Eps, MinPts$)
 1  $ClusterId = 1$
 2  for each point $u \in S$
 3      if $u.label \neq$ NIL
 4          continue
 5      $u.neighbor =$ REGION-QUERY($u, Eps$)
 6      if $\|u.neighbor\| < MinPts$
 7          $u.label =$ NOISE
 8      else
 9          EXPAND-CLUSTER($u, ClusterId, Eps, MinPts$)
10          $ClusterId = ClusterId + 1$
```

```
EXPAND-CLUSTER($u, ClusterId, Eps, MinPts$)
1  $u.label = ClusterId$
2  for each point $v \in u.neighbor$
3      if $v.label ==$ NIL
4          $v.label = ClusterId$
5          $v.neighbor =$ REGION-QUERY($v, Eps$)
6          if $\|v.neighbor\| < MinPts$
7              EXPAND-CLUSTER($v, ClusterId, Eps, MinPts$)
8      elseif $v.label ==$ NOISE
9          $v.label = ClusterId$
```

Similiar with DFS traverse, DBSCAN clustering is also divided into two procedures, DBSCAN and EXPAND-CLUSTER. DBSCAN functions as a wrapper function in the same way as DFS. This procedure goes through each point in the data set S. If the current point u has been visited, the procedure skips it. Otherwise, further process continues. However, unlike calling DFS-VISIT on every unvisited u unconditionaly as DFS does, in DBSCAN, another procedure EXPAND-CLUSTER is called if and only if u has sufficient number of neighbors in a given range Eps. REGION-QUERY(u, Eps) returns all the neighbors of u with a distance no further than Eps. This is the extra condition mentioned earlier. Similiarly, this condition is also checked in EXPLAND-CLUSTER at line 6, as opposed to DFS-VISIT. Besides these two places, the rest of the two algorithms are the same. Details of extra conditions are fully explained in next section.

3.2.2 Notions of Density in Clusters

From the view of density, points in a space can be classified into three categories: core points, border points, and outliers/noise. The classification criterion on a point u bases on the size of its *Eps-neighborhood*, denoted as $N(u; Eps)$. The $N(u; Eps)$ represents the collection of all points whose distances to u are within Eps, which is specified by user. More formally, $N(u; Eps) = \{v \mid dist(u, v) < Eps\}$.

Based on above definition, a point u is a core point if and only if $\|N(u; Eps)\| \geq MinPts$, where both Eps and $MinPts$ are user specified. After defining core points, it is relatively easy to define the other two categories. A border point v is a neighbor point of a core point u, but v is not a core point itself. Formally, $\|N(u; Eps)\| < MinPts$, and $v \in N(u; Eps)$, where $\|N(u; Eps)\| \geq MinPts$. For a outlier, it's a point v which is neither a core point itself nor a neighbor point of a core point. An example graph is showing in Fig 3.3.

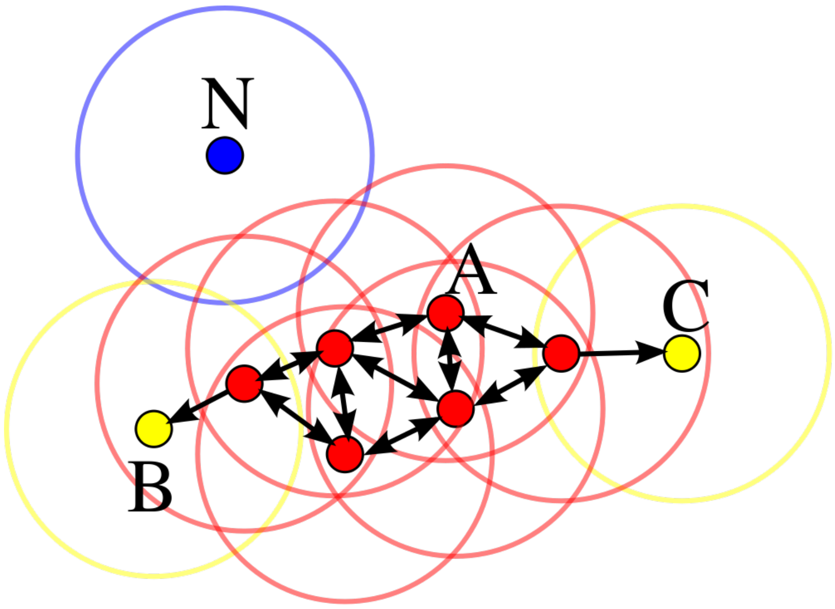

Figure 3.3: Illustration of core points(red), border points(yellow), and outlier(blue) [23]. In the illustration, $MinPts = 4$, Eps is the radius of those circles.

As show in the figure, red points are core points since in each red circle, there are at least 4 points(including the center point). While for the yellow point, they don't have enough points in their circles, but they embed in one of the red circles. Thus, they are border points. As for the blue point, it has neither enough point in the blue cirle nor is in any red circle. Thus, the blue point an outlier/noise.

These definitions reveal the assumption and theory of DBSCAN. DBSCAN assumes that, each cluster consists of two types of points, core points and border points. Core points form inner denser areas compared to spaces outside of clusters. Border points form a slightly sparser areas surrouding the inner core ares, but still the density is higher than spaces between different clusters, possesed by noise. In DBSCAN, the density of an area is measured by the number of points centered at u within an area specified by the given radius Eps. If there are at least $MinPts$ points, this area is considered to be a dense area. Thus, this area is part of a cluster. In the procedure EXPAND-CLUSTER, the algorithm will explore the whole dense area and less dense border area belonging to the same cluster. Each call of EXPAND-CLUSTER visits a different cluster. Thus, when the whole procedure DBSCAN finishes, different clusters are formed.

3.2.3 Analysis of DBSCAN

DBSCAN applies to a broader range of problems comapred to K-Means. In K-Means, one of the obstacle is to compute the mean value. DBSCAN doesn't have this problem as long as the distance between points is computable. This enables DBSCAN to use more compliated distance measurement, such as edit distance, which is very useful in dealing with strings. Another advantage of DBSCAN is its robustness to noises. As mentioned in section 3.1.2, the mean value will be heavily interfere by noises/outliers. In contrast, DBSCAN has a notion noises and it is able to spot these outliers, which is a core problem of this thesis. Finally, the most important advantage of DBSCAN is that it can find arbitrarily shaped clusters. It can even seperate a cluster completely surrounded by a different cluster. An example is shown in Fig 3.4.

Simliar to K-Means, DBSCAN also requires user specified parameters. To get the optimal result, a careful choice of these parameters is needed. Finding the appropriate parameter can be achieved in the similiar way by finding the elbow point as mentioned in section 3.1.2. The user can pick a number for $MinPts$ first. Then, for each point, the distance from its kth nearest neighbor is computed. After sorting these distances in descending order and plot them, a graph called *sortedk-distgraph* can be obtained. This graph reveals insights about the density distribution of the whole dataset reflected in how the *k-dist* varies. Then the Eps can be set to the value corresponding to the elbow point. This heuristic works well as the graph won't differ too much for $k > 4$. An illustration of this approach is shown in Figure 3.5. [11]

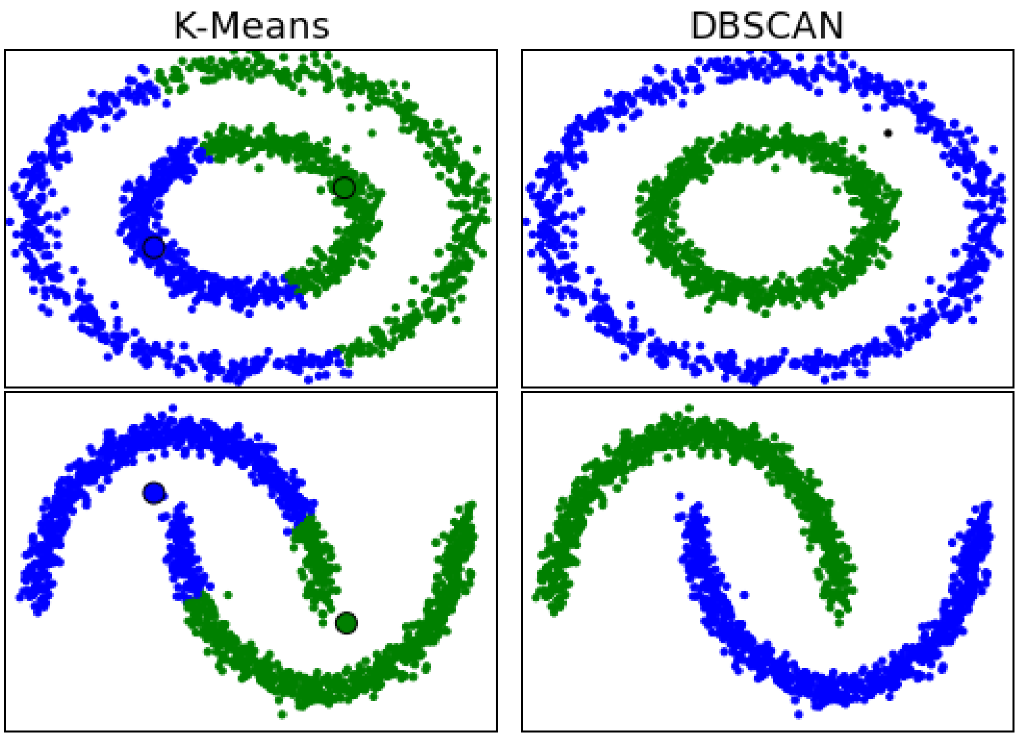

Figure 3.4: A Comparison of K-Means and DBSCAN. In the diagram, two datasets "circles" and "moons" are used. Compared to K-Means, DBSCAN gives more reasonable clustering results on these two irregular shaped datasets.

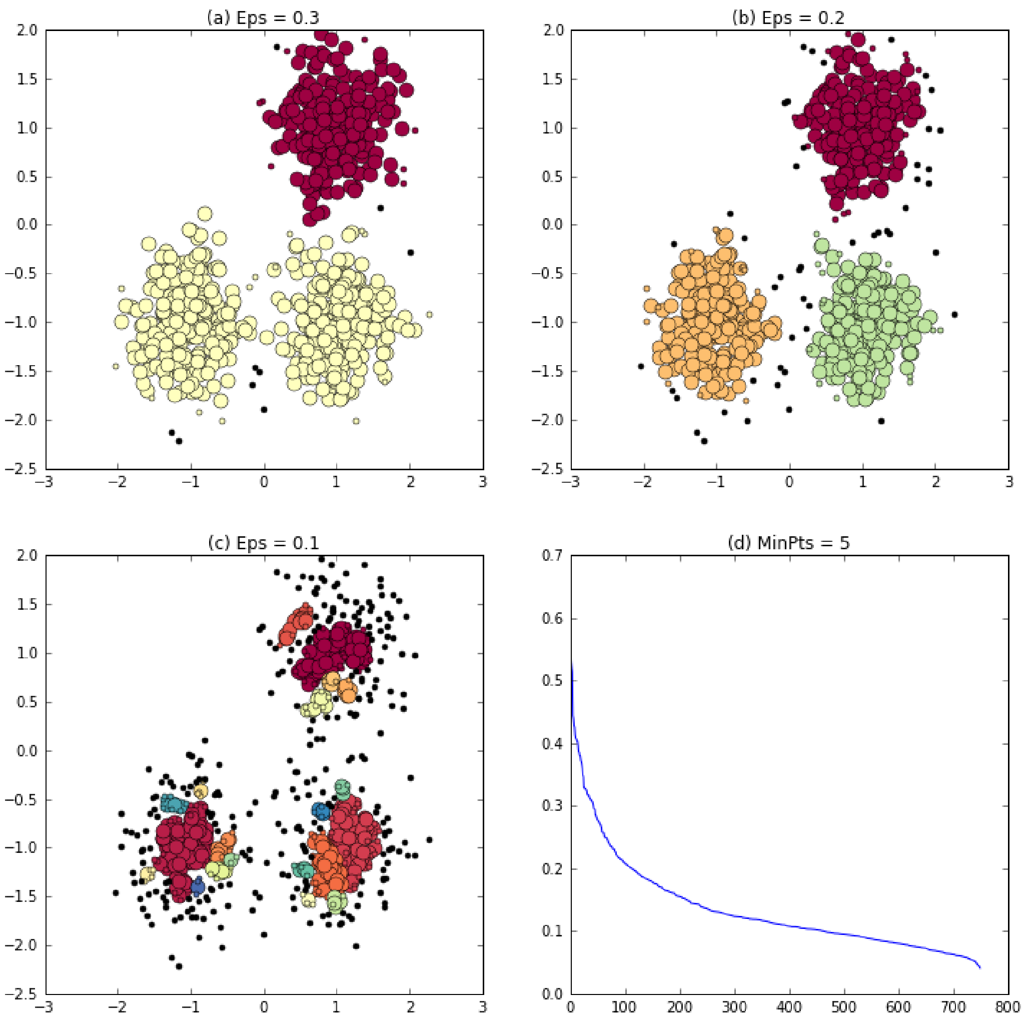

Figure 3.5: Illustration of the heuristic parameter choosing approach for DBSCAN. (a)-(c) illustrates clustering results obtained by using different *Eps*. Each cluster is painted with a different color. Core points are represented using larger circles while border points are represented using smaller circles. Black small circles represent outliers. In figure (a), very few points are labeled as noise and the bottom two clusters are not distinguished. In figure (b), the result is more reasonable and can be considered as optimal. In figure (c), too many points are labeled as noise and more than 3 clusters are reported. According to the *k-distgraph* shown in (d), 0.2 should be the best value for *Eps*, which corresponds to the result in figure (b).

Chapter 4

Generative Methods

This chapter is going to explore generative methods for anomaly detection. Generative methods are a collection of algorithms which try to build a model that explains the process of how the data generates. Then, based on information from the proposed model, anomalies are detected. A family of algorithms belonging to this category is the Markov models.

As described in Chapter 2, the patients visits can be seen as time sequential data consisting of a series of events. The events in one visit are not generated independently and randomly. Instead, past events have an effect on the type of the next possible event. To handle sequential data, Markov models are the correct choice since they consider the relation between consecutive observations. In the following context, Section 4.1 introduces the basic Markov chain model. Later, Section 4.2 expands the Markov chain to a more complicated Hidden Markov Model by introducing hidden variables.

4.1 Discrete Markov Process

Consider a system having N distinct states $S_1, S_2, \cdots, X_N$. At any time, the system will be in one of these states. After a given time period t, a series obeservation $x_1, x_2, \cdots, x_t$ can be obtained. (Without loss of generality, the following discussion assumes the variables are all scalar. The assumption holds in the rest of the context unless explicitly stated otherwise) According to the prodcut rule of probability, the joint probability distribution for this sequence of observations is

$$p(x_1, x_2, \cdots, x_t) = \prod_{i=2}^{t} p(x_i \mid x_1, \cdots, x_{i-1}) \tag{4.1}$$

The conditional probability distribution of each observation x_i depends on all observations having a smaller index than it. The above relations between the observations can be represented graphically in Fig 4.1(a). The graph is fully connected, and no independence property can be obtained from it. Now assume that each observation x_i only depends on one immediate previous observation x_{i-1}. Then the joint distribution becomes

$$p(x_1, x_2, \cdots, x_t) = p(x_1) \prod_{i=2}^{t} p(x_i \mid x_{i-1}) \tag{4.2}$$

This newly obtained model is depicted in Fig 4.1(b), and is referred as *first-order Markov chain*. The term *first-order* indicates the dependence on only one previous observation. Suppose the system has only 3 states, as shown in Fig 4.1(c). Then, to fully represent the system, the only required information is the transition probabilities between different sates. The transition probabilities are usually refered as *transition matrix*, denoted as $\mathbf{A}$. Each element A_{ij} represents the probability of transfering from state s_i to state s_j. Learning the parameters of this model is very simple. Since the states are exactly the observations, A_{ij} can be simply obatained by compute the frequency of transfering to s_j starting from s_i. The number of free parameters in this model is $K(K-1)$, where K represents the number of states in the system.

Sometimes, the observations can depend on more than one observations in the past. One simple way to achieve this is creating a higer order Markov chain. By allowing each observation to depend on previous two values, a second-order Markov chain is obtained, as shown in Fig 4.2. Then the joint distribution becomes

$$p(x_1, x_2, \cdots, x_t) = p(x_1)p(x_2 \mid x_1) \prod_{i=3}^{t} p(x_i \mid x_{i-1}, x_{i-2}) \tag{4.3}$$

Compared to *first-order Markov chain*, the *second-order Markov chain* has better capability of modelling complex relations between varaibles. In fact, the higher the order is, the more flexible the model is. However, the number of parameters grows as well, which makes the model difficult to train. For a M^{th} order markov chain, there will be $K^M(K-1)$ parameters. Because the exponential growth in number of parameters, the model gardually becomes impractial as M grows.

The markov chain model can be expanded in another way, by associating an emission distribution $\mathbf{E}_k$, $k = 1, \cdots, K$, to each state in the system. Thus, two observations x_i, y_i exist at any time, where y_i is generated depending on x_i. If the expansion is based on a first-order Markov chain, the joint

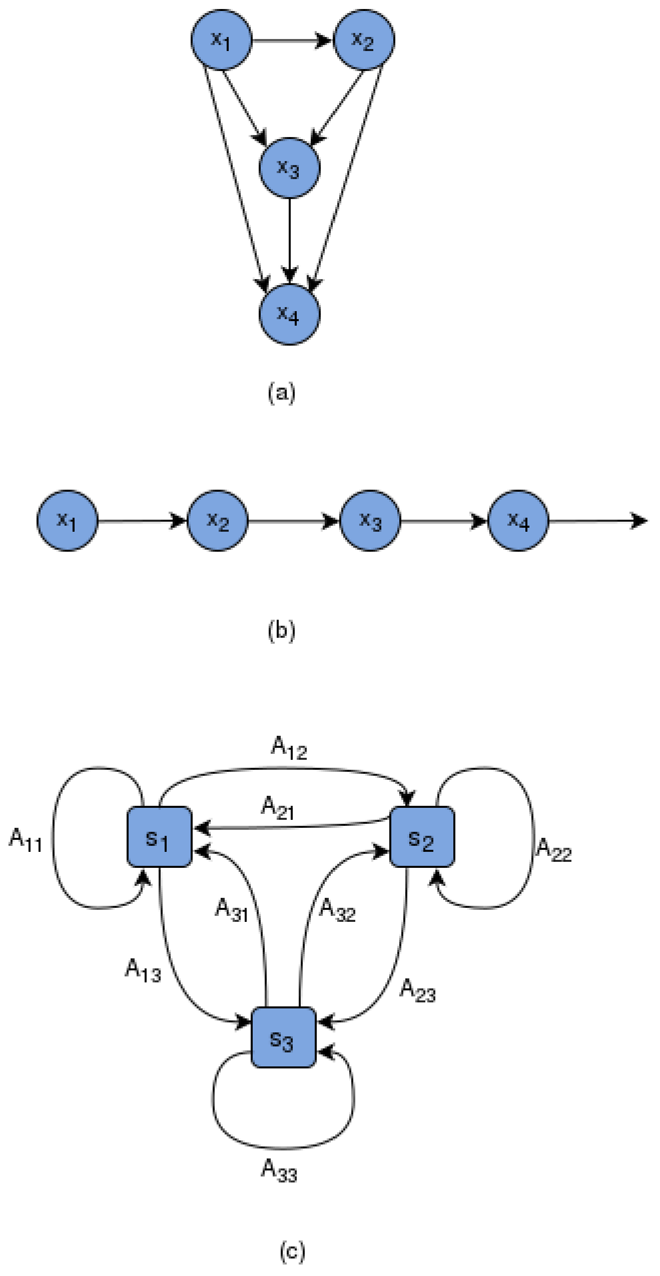

Figure 4.1: Illustration of a Markov Chain of 4 observations possessing 3 states. Variables are represented using filled circles, while states are represented using filled squares.

distribution becomes

$$p(x_1, y_1, \cdots, x_t, y_t) = p(x_1)\Big(\prod_{i=2}^{t} p(x_i \mid x_{i-1})\Big)\Big(\prod_{i=1}^{t} p(y_i|x_i)\Big) \tag{4.4}$$

The patient visits can be seen as an example of this expanded model. For each visit, the patient will go through a series of events, which can be considered as states. Each event will then last for a certain period of time. The duration can be seen as generated from a distribution, and the parameters of this distribution depend on the event. The example is shown in Fig 4.3.

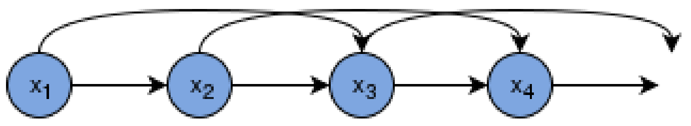

Figure 4.2: Illustration of a second-order Markov model.

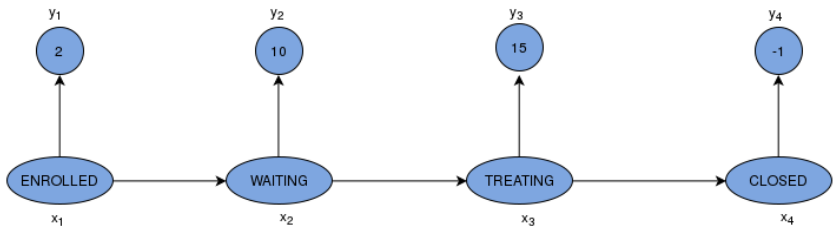

Figure 4.3: Modelling a patient visit as an expansion of first-order Markov model.

4.2 Hidden Markov Models

As mentioned in the last section, a tradeoff between flexibility and practicality exists in deciding the order number for a Markov chain model. In special situations, such as speech recognition, very high-order Markov chain is expected. But the computation resource requires as few number of parameters as possible. It would be ideal if a model is not limited to any specific given order, and still only limited number of parameters are required to specify the model. Luckily, these requirements can be satisfied by constructing a Hidden Markov Model using additional latent variables.

Suppose a sequence of observations $\mathbf{X} = \{x_1, \cdots, x_N\}$ is obtained. Instead of assuming each observation depends directly on a specific number of previous observations, the new assumption is that, there is a latent variable

z_i corresponding to each observation, and the latent variables form a Markov chain. The latent variables don't have to possess any physical meanings. They can even be of different type to the observations, in terms of distribution and dimensionality. A graphical representation of this model is shown in Fig 4.4. It's easy to get confused by comparing Fig 4.3 and Fig 4.4 since they share the same graphical structure. The difference is that, in Fig 4.4, the z_i's are unobserved latent variables, which is depicted using unfilled circles. While in Fig 4.3, both events and duration are observed values. All observed variables are represented using filled circles. If the HMM is applied to the patient visits, one potential structure could be the one shown in Fig 4.5. In this model, the value of latent variable determines which event will generate, then the event determines how long the duration will be. Notice that the latent variables don't have any associated physical meaning or specific distribution form. One can explain them as indication of the functioning statu of the system by selecting them to be binary variables. When $z_i = 1$, it indicates the queue system in the hospital is working in normal mode. When $z_i = 0$, it means the system is working in a problametic way.

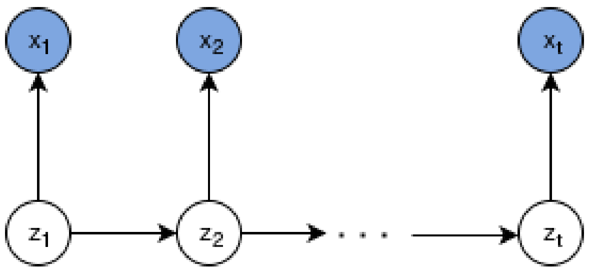

Figure 4.4: Graphical representation of a Hidden Markov Model. Observations are represented using filled circles, whilc latent variables are depicted using unfilled circles. The latent variables form a first-order Markov chain.

In the framework of HMM, the joint distribution over both observed and latent variables is given below

$$p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) = p(z_1 \mid \pi)\Big(\prod_{n=2}^{N} p(z_n \mid z_{n-1}, \mathbf{A})\Big)\Big(\prod_{m=1}^{N} p(x_m \mid z_m, \phi)\Big) \tag{4.5}$$

where $\mathbf{X} = \{x_1, \cdots, x_N\}$ represents all the observed variables, $\mathbf{Z} = \{z_1, \cdots, z_N\}$ represents latent variables, and $\boldsymbol{\theta} = \{\pi, \mathbf{A}, \phi\}$ represents the parameters in this model. The π is a prior distribution for deciding the value of the first variable z_1. The matrix $\mathbf{A}$ is the transition matrix among the latent vari-

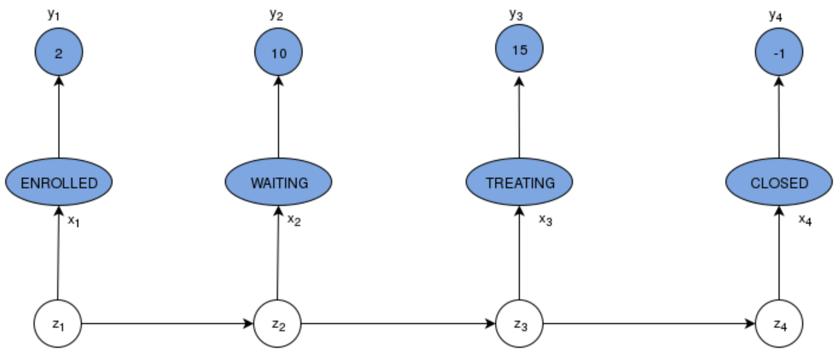

Figure 4.5: Modelling patient visit as Hidden Markov Model. Event types and event duration are represented using filled eclipses and circles respectively. Additional hidden variables are represented using unfilled circles. No specific physical meaning is associated with these latent variables.

ables. The ϕ are the parameters of the emission distribution associated with z_i and x_i.

4.2.1 Learning and Inference

There are three basic problems in HMM. [22] These problems are described below using above notations:

- Problem 1: Given a sequence of observations $\mathbf{X} = \{x_1, \cdots, x_N\}$, what is the probability $p(\mathbf{X} \mid \boldsymbol{\theta})$over the observations, under specific parameters $\boldsymbol{\theta} = \{\pi, \mathbf{A}, \phi\}$?
- Problem 2: What's the value of the parameters which maximizes $p(\mathbf{X} \mid \boldsymbol{\theta})$?
- Problem 3: Given a sequence of observations $\mathbf{X} = \{x_1, \cdots, x_N\}$, what is the value of the corresponding latent variables?

The rest of the context focus on the first two questions. The reason is that, once the value of $p(\mathbf{X} \mid \boldsymbol{\theta})$ is computed, the decision on whether a given sequence is anomaly can be made by comparing $p(\mathbf{X} \mid \boldsymbol{\theta})$ to a threshould value.

Though it seems more intuitive that finding a way to evaluate $p(\mathbf{X} \mid \boldsymbol{\theta})$ should come before maximizing it with respect to the parameters, it would be more convenient to start at solving problem 2. After solving problem 2, the solution to the first problem will appear naturally. The following discussion begins by introducing some new concepts and notations.

The distribution over only observed variables $p(\mathbf{X} \mid \boldsymbol{\theta})$ is usually refered as *incomplete likelihood*, while distribution over both observed and unobserved variables $p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$ is refered as *complete likelihood.* Using formula (4.5), the logarithm of incomplete likelihood can be represented as

$$\begin{aligned}\ln p(\mathbf{X} \mid \boldsymbol{\theta}) &= \ln \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \\ &= \ln p(z_1 \mid \pi) + \ln \sum_{\mathbf{Z}} \Big(\big(\prod_{n=2}^{N} p(z_n \mid z_{n-1}, \mathbf{A}) \big) \big(\prod_{m=1}^{N} p(x_m \mid z_m, \phi) \big) \Big)\end{aligned} \tag{4.6}$$

The above formula is a generalization of the *mixture distribution* [7]. Maximizing this formula with respect to the parameters is very difficult since the derivatives don't have a closed form. An alternative practical working algorithm is the *expectation-maximization(EM)* algorithm [10][20]. The EM algorithm is very similiar to the K-Means algorithm mentioned in Chapter 3. The algorithm consists of two steps, E-step and M-step. In the E-step, the algorithm fixes the value of parameters and find the posterior distribution of the laten variables $p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{old})$. Here the notation is adopted from Bishop's [7]. The superscription *old* in θ^{old} means the parameter is fixed. Then the algorithm computes the expectation of the logarithm of the complete likelihood, with respect to the derived posterior distribution. The newly derived term becomes a function of $\boldsymbol{\theta}$, which is shown below

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) = \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}^{old}) \ln p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta}) \tag{4.7}$$

Then in the M-step, the new value of $\boldsymbol{\theta}$ is updated by maximizing $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old})$. Compared to K-Means, the E-step corresponds to assign each point to a cluster prototype, and the M-step corresponds to update the value of the prototypes. These two steps are excecuted alternatively until convergence or maximum number of iteration is reached. In the rest context, $\gamma(\mathbf{z}_n)$ and $\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n)$ are introduced which stands for the posterior distribution of a single latent variable and the joint posterior distribution over two consecutive latent variables, seperately. Instead of assuming the latent variables are scalar, here they are represented using *1-of-K* coding. Namely, each latent variable is a length K vector, where one and only one of these K elements equals 1. When $z_{nk} = 1$, it means the *nth* latent variable is in the *kth* state.

Using this representation schema, following equations are obtained

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{old}) \tag{4.8}$$

$$\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}, \boldsymbol{\theta}^{old}) \tag{4.9}$$

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{old}) =& \sum_{k=1}^{K} \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^{N}\sum_{j=1}^{K}\sum_{k=1}^{K} \gamma(\mathbf{z}_{n-1}, \mathbf{z}_n) \ln A_{jk} \\ &+ \sum_{n=1}^{N}\sum_{k=1}^{K} \gamma(z_{nk}) \ln p(x_n|\phi_k) \end{aligned} \tag{4.10}$$

Computation in the M-step is relatively easy. Assume the E-step has been done, so that $\gamma(\mathbf{z}_n)$ and $\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n)$ are like constants now. Then following update equation can be obtained

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})} \tag{4.11}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \gamma(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K}\sum_{n=2}^{N} \gamma(z_{n-1,j}, z_{nl})} \tag{4.12}$$

Updation of ϕ_k is more tricky, since it depends on the specific choice of the emission distribution. One good observation is that, only the final term depends on ϕ_k, and different ϕ_k doesn't couple with each other. Thus, each ϕ_k can be updated seperately. The term $\gamma(z_{nk})$ functions as a soft assignment, representing the probability of assigning a point x_n to each state.

Computation in E-step is more difficult which requires efficient algorithm. The most widely used algorithm is known as *alpha-beta* algorithm. This algorithm can be seen as an application of dynamic programming technique which takes advantage of the tree structure in HMM thus leading to efficiency. To start with the alpha-beta algorithm, following conditional independence properties should be obtained first [16]

$$\begin{aligned} p(\mathbf{X}|\mathbf{z}_n) =& p(x_1, \cdots, x_n|\mathbf{z}_n) \\ & p(x_{n+1}, \cdots, x_N|\mathbf{z}_n) \end{aligned} \tag{4.13}$$

$$p(x_1, \cdots, x_{n-1}|x_n, \mathbf{z}_n) = p(x_1, \cdots, x_{n-1}|\mathbf{z}_n) \tag{4.14}$$

$$p(x_1, \cdots, x_{n-1}|z_{n-1}, \mathbf{z}_n) = p(x_1, \cdots, x_{n-1}|\mathbf{z}_{n-1}) \tag{4.15}$$

These equations can be obtained by using *d-seperation* technique [21], or proved formally using sum and product rules of probability. Using the first independence property and Bayes' theorem, following equations are obtained

$$\begin{aligned}\gamma(\mathbf{z}_n) = p(\mathbf{z}_n|\mathbf{X}) &= \frac{p(\mathbf{X}|\mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{p(x_1, \cdots, x_n, \mathbf{z}_n)p(x_{n+1}, \cdots, x_N|\mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}\end{aligned} \tag{4.16}$$

where

$$\alpha(\mathbf{z}_n) = p(x_1, \cdots, x_n, \mathbf{z}_n) \tag{4.17}$$

$$\beta(\mathbf{z}_n) = p(x_{n+1}, \cdots, x_N|\mathbf{z}_n) \tag{4.18}$$

Using the other two conditional independence properties, $\alpha(\mathbf{z}_n)$ can be expressed recursively in terms of $\alpha(\mathbf{z}_{n-1})$

$$\begin{aligned}\alpha(\mathbf{z}_n) &= p(x_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}} p(x_1, \cdots, x_{n-1}, \mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1}) \\ &= p(x_n|\mathbf{z}_n)\sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1})p(\mathbf{z}_n|\mathbf{z}_{n-1})\end{aligned} \tag{4.19}$$

Similiarly, $\beta(\mathbf{z}_n)$ can also be expressed recursively as

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1})p(x_{n+1}|\mathbf{z}_{n+1})p(\mathbf{z}_{n+1}|\mathbf{z}_n) \tag{4.20}$$

The term $\alpha(\mathbf{z}_n)$ can be seen as messages propagated from the beginning to the end. Each $\alpha(\mathbf{z}_n)$ receives messages passed from its predecessor, combines these information with its own information and then pass them to its successor. The logical also applies to the term $\beta(\mathbf{z}_n)$, but the messages are from the end to the beginning. Due to the tree strucutre in HMM, computing each term only depends on one adjacent term, instead of all terms before/after it. Thus, the computation reduces dramatically which makes the algorithm efficient. To start the whole computation, initial conditions $\alpha(\mathbf{z}_1)$ and $\beta(\mathbf{z}_n)$ are required. The initial conditions are given below

$$\alpha(\mathbf{z}_1) = \prod_{k=1}^{K}\{\pi_k p(x_1|\phi_k)\}^{z_{1k}} \tag{4.21}$$

$$\beta(\mathbf{z}_N) = 1 \tag{4.22}$$

Having obtained $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$, the posterior distribution $\gamma(\mathbf{z}_n)$ can be computed as in equation (4.16). As for $\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n)$, it can be computed as following

$$\begin{aligned}\gamma(\mathbf{z}_{n-1}, \mathbf{z}_n) &= p(\mathbf{z}_{n-1}, \mathbf{z}_n|\mathbf{X}) \\ &= \frac{p(\mathbf{X}|\mathbf{z}_{n-1}, \mathbf{z}_n)p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})} \\ &= \frac{p(x_1, \cdots, x_{n-1}|\mathbf{z}_{n-1})p(x_n|\mathbf{z}_n)p(x_{n+1}, \cdots, x_N|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})p(\mathbf{z}_{n-1})}{p(\mathbf{X})} \\ &= \frac{\alpha(\mathbf{z}_{n-1}p(x_n|\mathbf{z}_n)p(\mathbf{z}_n|\mathbf{z}_{n-1})\beta(\mathbf{z}_n)}{p(\mathbf{X})} \qquad (4.23)\end{aligned}$$

Up till now, both steps in EM algorithm are introduced, and the problem 2 can be solved efficiently. The left question is how to solve problem 1, computing the likelihood over the incomplete data. The solution comes from equation (4.16). Notice that $\gamma(\mathbf{z}_n)$ is a posterior distribution. Integrating both sides of equation (4.16) over $\mathbf{z}_n$ gives

$$p(\mathbf{X}) = \sum_{\mathbf{z}_n} \alpha(\mathbf{z}_n)\beta(\mathbf{z}_n) \qquad (4.24)$$

where $\mathbf{z}_n$ is an arbitrary latent variable. If $n = N$, then $\beta(\mathbf{z}_n) = 1$, which makes the above equation simpler

$$p(\mathbf{X}) = \sum_{\mathbf{z}_N} \alpha(\mathbf{z}_N) \qquad (4.25)$$

Then, both problem 1 and problem 2 are solved.

Chapter 5

Experiment

This chapter describes the detailed experimental implementation. Following topics will be discussed:

- Which specific data are used? How to represent the data? What pre-process has been done? What related functions are defined?
- Which methods are used? What's the reason to use such methods?

The three topics will be discussed in three sections, respectively. Analysis of the result will be postponed to Chapter 6.

5.1 Data Details and Representation

Since the akseli system has gone through several updation after its first release, many features have changed, including how the visit log is reocrded. Considering this aspect, only data generated after year 2014 is used in the experiment. All the data are from Oulu Hospital, with patient privacy information eliminated. In total, 243K unique visits with 1.93 million entries are retrieved, which spans from January to July. The retrieved data consists of three columns: visit id, event type, and recorded time of the event. Other information, such as which resource generated the data, is not retrieved. Among the retrieved data, 7 event types are adopted. The used event types are: `ENROLLING`, `WAITING`, `IN_TREATMENT_ROOM`, `PAUSED`, `IN_TREATMENT_ROOM_FROM_PAUSED`, `CLOSED`, and `CANCELLED`.

Huang et al [**?**] proposed to represent the patient visits as a sequence of pairs. Each pair contains the information of (a). what the event is and (b). when this event happened. For example, given a sequence showing below

$$\langle (a, 1), (b, 2), (c, 5), (d, 7) \rangle$$

it means a patient comes to the hospital, and at time 1, the patient encounters event a. Then at time 2, the patient encounters event b, and so on. The time unit can be selected arbitrarily, such as minutes, hours, or days. Huang et al used days as the time unit. In their work, they tried to cluster the patient traces. However, the patient trace has different length. Thus, typical distance metrics such as euclidean distance is not applicable. To address this problem, they also proposed a new distance metric, based on edit distance.

Edit distance is commonly used in comparing strings and biological sequences, such as proteins. Edit distance is defined as the minimum number of allowd operations used, to transform a string s to another string t. For example, if the allowed operations are *delete*, *insert*, and two strings $S =$ *"array"*, $T =$ *"xray"* are given. Then the edit distance between s and t is 3, by taking 3 operations. One potential transformation is:

1. Delete the second letter r in S by x. Then S becomes *"aray"*.
2. Delete the first letter a in S. Then S becomes *"ray"*.
3. Insert letter x at the beginning of S. Then S becomes *"xray"*, which is the same with string T.

The edit distance problem can be solved effectively by using the dynamic programming technique. Using terminology from dynamic programming, the optimal solution to the edit distance problem can be represented recursively

$$D(i,j) = \begin{cases} D(i-1, j-1) & \text{if } S[\text{i}] = T[\text{j}] \\ \min\{D(i-1,j), D(i,j-1)\} + 1 & \text{if } S[\text{i}] \neq T[\text{j}] \end{cases}$$

The edit distance only considers the difference between types of events, when applied to the patience visit data. However, the time associated with each event should also makes an effect when comparing two traces. Huang et al addressed this problem by providing a modified edit distance. In the old edit distance, events from two patient trace will either increase the distance by 1 if they belong to different type or 0 if they belong to the same type. In the modified distance, however, the increament caused by two events range from $[0, 1]$ as shown below

$$\delta(\sigma_i, \rho_j) = \begin{cases} 1 & \text{if } \sigma_i(e) \neq \rho_j(e) \\ \frac{|\sigma_i(t) - \rho_j(t)|}{max\{\sigma_i(t),\ \rho_j(t)\}} & \text{if } \sigma_i(e) = \rho_j(e) \end{cases}$$

where σ and ρ are two patient traces. σ_i is the ith pair of the trace. $\sigma_i(e)$ and $\sigma_i(t)$ represent the event type and timestamp of the ith pair in that trace. The intuition of the above formula is that, if the event types of two pairs in two traces are different, then they contribute 1 to the edit distance. If the event types are the same, then the distance is determined by the timestamp associated with the two events. The closer the timestamps are, the smaller the distance is.

The modified edit distance seems reasonable. However, some subtle issues exist when the modified edit distance applies to Huang's representation. Consider two patient traces

$$S = \langle (a,1),(b,1000),(c,1001),(d,1002)\rangle$$
$$T = \langle (a,1),(b,2),(c,3),(d,4)\rangle$$

The two traces are very similiar, except that the second pair differs greatly. But this difference propagates further to the third and fourth pairs, incurring more penalty. The modified edit distance will equal almost 3. It would be more reasonable if the distance accounts only the huge different generated in the second pairs, and considers the third and fourth pair the same. To address this problem, in the experiment, a modification on the representation form is applied. Rather than record the absolute timestamp associated with each event, the duration of each event is recorded. Thus, the above two patient traces becomes

$$S = \langle (a,1),(b,999),(c,1),(d,1)\rangle$$
$$T = \langle (a,1),(b,1),(c,1),(d,1)\rangle$$

Applying the modified edit distance to the new representation, the answer equals roughly 1, which is more intuitive. Thus, in all experiments, the second representation form is adopted. Minute is used as the time unit.

Another critical point is about preprocessing. After using the second representation form, numerous noise points are observed. It's believed that the noise points are generated by the system itself for logging reasons. The feature of the noise is that all events have a 0 duration time. The noise points consist of approximately 20% of all data, which incurs great effect in training models. Thus, in the preprocess step, all noise data are manually removed.

5.2 Methods

Chapter 3 and chapter 4 introduced 4 potential methods. However, only DBSCAN and Markov Chain are used in the experiment. This section explains the reasons for choosing only these two methods and describe related details.

5.2.1 Choice of Clustering Method

As stated in Chapter 3, both DBSCAN and K-Means need a elaborate distance metric. This requirement can be fullfilled by using the modified edit distance. Besides this, K-Means also requires efficient computation of the mean value. However, based on the current data representation, it is not clear how to compute the mean. Thus, K-Means is not applicable in this setup.

One critical phase in applying DBSCAN is how to efficiently compute the *Eps-neighborhood*. The DBSCAN algorithm will go through several iterations. Each iteration involves computing *Eps-neighborhood* for all points. The efficiency of computing *Eps-neighborhood* directly determines the practicability of DBSCAN. A naive implementation is to compute a N by N pair-wise distance table. Then sort the rows in the matrix. After this, finding the *Eps-neighborhood* takes only constant time. This process takes $O(N^2)$ space and time. The space complecity can be further reduced to $O(N)$.

A more efficient method, vantage-point tree [**?**] can be used in the experiment. Vantage-point tree is a recursively built balanced binary tree. Each non-leaf node consists of two fields, a point functions as a center and a radius, and has two children. The center point is randomly selected from a set of available points. Initially, the set is the whole data points. Then, distances from the center to all the rest points in the set is computed. Next, the radius is set to the median of the distances. After this, the point set is divided into two subsets, one consists of points with a distance shorter than the radius, the other consists of points with a distance longer than the median value. The first set is stored in the left child of current node, and the second set is stored in the right child of the current node. Intuitively, this works as drawing a circle centered on the selected center point. The circle partitions the other points into two parts, with half inside the circle, half outside the circle. This process continues until a leaf node is encountered. This building process takes $O(N \log N)$ time and $O(N \log N)$ space.

After building the vantage-point tree, finding the *Eps-neighborhood* for a single point takes $O(\log N)$ time. Suppose the query point is p. Now the query enters a node centered on point q and the radius of the node is τ. Suppose the distance from p to q is $\delta < \tau$. Then the query explores only the left child of q if $\delta + \epsilon \leq \tau$. Otherwise, the right child is also explored. The intuition is that, if one *Eps-neighborhood* of query point p is exactly ϵ far to p, then the distance between this neighbour point and the centerd point q is at most $\delta + \epsilon$. If this distance is no larger than the radius associated to the centered node, then there is no need to explore points outside the circle, which reduces the search space by half. As a result, find *Eps-neighborhood* for all points takes $O(N \log N)$ time.

5.2.2 Choice of Generative Method

In the experiment, the simple Markov Chain model is selected for several reasons. Firstly, the patient trace is not long. A typical patient trace has only four events, `ENROLLING`, `WAITING`, `IN_TREATMENT_ROOM`, and `CLOSED`. The longest patient trace doesn't have more than 10 events. Even building an explicit high-order Markov Chain model on these data will still be practical. Another point is that, the type of potential event closely depends on the previous one, rather than depending on an implicit statu. For example, after the event `ENROLLING`, it is very likely the next event is `WAITING`. Sometimes, due to special situations such as cancellation of the reservation, the event `CLOSED` follow. But it is impossible a `IN_TREATMENT_ROOM` event comes, which skips the `WAITING` phase. The last argument is that, anomaly happened in one phase does not necessarily affect the coming phase. For example, a patient may encounter problems while being in the `WAITING` phase. But this does not mean the patient may also encounter problems while in `IN_TREATMENT_ROOM` phase. In other words, being in abnormal status in each phase is independent, which is quite contrary to the assumption of HMM that the value of current hidden variable depends on value of the previous hidden variables. As a result, a simple one-order Markov Chain model is selected in the experiment.

In the one-order Markov Chain model, the events are represented using *1-of-K* coding schema.

Chapter 6

Evaluation

You have done your work, but that's[1] not enough.

You also need to evaluate how well your implementation works. The nature of the evaluation depends on your problem, your method, and your implementation that are all described in the thesis before this chapter. If you have created a program for exact-text matching, then you measure how long it takes for your implementation to search for different patterns, and compare it against the implementation that was used before. If you have designed a process for managing software projects, you perhaps interview people working with a waterfall-style management process, have them adapt your management process, and interview them again after they have worked with your process for some time. See what's changed.

The important thing is that you can evaluate your success somehow. Remember that you do not have to succeed in making something spectacular; a total implementation failure may still give grounds for a very good master's thesis—if you can analyze what went wrong and what should have been done.

[1]By the way, do *not* use shorthands like this in your text! It is not professional! Always write out all the words: "that is".

Chapter 7

Discussion

At this point, you will have some insightful thoughts on your implementation and you may have ideas on what could be done in the future. This chapter is a good place to discuss your thesis as a whole and to show your professor that you have really understood some non-trivial aspects of the methods you used…

Chapter 8

Conclusions

Time to wrap it up! Write down the most important findings from your work. Like the introduction, this chapter is not very long. Two to four pages might be a good limit.

Bibliography

[1] Aalto University, Helsinki University of Technology, Committee on Academic Affairs. Guidelines for master's thesis evaluation, December 4 2008. `http://www.tkk.fi/fi/opinnot/opintohallinto/paatokset/guidelinemastersthesisevaluation.pdf` (in English), `http://www.tkk.fi/fi/opinnot/opintohallinto/paatokset/diplomityoarviointiohje.pdf` (in Finnish).

[2] Aalto University Library. webpage, March 2 2011. How to find - brief guide to finding publications and information. Accessed 4.3.2011.

[3] Aalto University Library. Making a bibliography. webpage, March 2 2011. `http://otalib.aalto.fi/en/instructions/guides/citations/bibliography/`. Accessed 4.2.2011.

[4] Aalto-yliopisto, Degree programme of Computer Science and Engineering. Instruction pages: Diplomityö, 2010. `http://information.tkk.fi/fi/opinnot/tietotekniikka/diplomityo/` (in Finnish), `http://information.tkk.fi/en/studies/cse/thesis/` (in English). Accessed 9.2.2011.

[5] Aloise, D., Deshpande, A., Hansen, P., and Popat, P. Np-hardness of euclidean sum-of-squares clustering. *Machine learning 75*, 2 (2009), 245–248.

[6] Becker, B., Gschwind, S., Ohler, T., Seeger, B., and Widmayer, P. An asymptotically optimal multiversion B-tree. *The VLDB Journal 5*, 4 (1996), 264–275.

[7] Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[8] Campello, R. J., Moulavi, D., Zimek, A., and Sander, J. Hierarchical density estimates for data clustering, visualization, and outlier detection. *ACM Transactions on Knowledge Discovery from Data (TKDD) 10*, 1 (2015), 5.

[9] Davis, J. C., and Sampson, R. J. *Statistics and data analysis in geology*, vol. 646. Wiley New York et al., 1986.

[10] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)* (1977), 1–38.

[11] Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd* (1996), vol. 96, pp. 226–231.

[12] Forsyth, D. A., and Ponce, J. *Computer vision: a modern approach.* Prentice Hall Professional Technical Reference, 2002.

[13] Haapasalo, T. *Accessing Multiversion Data in Database Transactions.* PhD thesis, Department of Computer Science and Engineering, Aalto University School of Science and Technology, Espoo, Finland, 2010. `http://lib.tkk.fi/Diss/2010/isbn9789526033600/`.

[14] Haapasalo, T., Jaluta, I., Seeger, B., Sippu, S., and Soisalon-Soininen, E. Transactions on the multiversion B^{+}-tree. In *Proceedings of the 12th Conference in Extending Database Technology* (2009), pp. 1064–1075.

[15] He, Z., Xu, X., and Deng, S. Discovering cluster-based local outliers. *Pattern Recognition Letters 24*, 9 (2003), 1641–1650.

[16] Jordan, M. I. An introduction to probabilistic graphical models, 2003.

[17] KDD Org. 2014 sigkdd test of time award. webpage, 2014. `http://www.kdd.org/News/view/2014-sigkdd-test-of-time-award`. Accessed 11.8.2016.

[18] Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory 28*, 2 (1982), 129–137.

[19] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA., pp. 281–297.

[20] McLachlan, G., and Krishnan, T. *The EM algorithm and extensions*, vol. 382. John Wiley & Sons, 2007.

[21] Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference.* Morgan Kaufmann, 2014.

[22] Rabiner, L. R. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE 77*, 2 (1989), 257–286.

[23] Wikipedia. DBSCAN — Wikipedia, the free encyclopedia. webpage, 2016. `https://en.wikipedia.org/wiki/DBSCAN`, Accessed 9.8.2916.

Appendix A

First appendix

This is the first appendix. You could put some test images or verbose data in an appendix, if there is too much data to fit in the actual text nicely.

For now, the Aalto logo variants are shown in Figure A.1.

Aalto University
School of Science

(a) In English

Aalto-yliopisto
Perustieteiden
korkeakoulu

(b) Suomeksi

Aalto-universitetet
Högskolan för
teknikvetenskaper

(c) På svenska

Figure A.1: Aalto logo variants