Modelling Patient States in Intensive Care Patients

by

Kanak Bikram Kshetri

S.B., Massachusetts Institute of Technology (2011)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Computer Science and Electrical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011

Author . . . .
Department of Electrical Engineering and Computer Science
August 30, 2011

Certified by . . . .
Prof. Peter Szolovits
Professor of Computer Science and Engineering
Thesis Supervisor

Certified by . . . .
Dr. Rohit Joshi
Postdoctoral Associate
Thesis Supervisor

Accepted by . . . .
Prof. Dennis M. Freeman
Chairman, Master of Engineering Thesis Committee

Modelling Patient States in Intensive Care Patients

by

Kanak Bikram Kshetri

Submitted to the Department of Electrical Engineering and Computer Science
on August 30, 2011, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Computer Science and Electrical Engineering

Abstract

Extensive bedside monitoring in hospital Intensive Care Units (ICU) has resulted in a deluge of information on patient physiology. Consequently, clinical decision makers have to reason with data that is simultaneously large and high-dimensional. Mechanisms to compress these datasets while retaining their salient features are in great need.

Previous work in this area has focused exclusively on supervised models to predict specific hazardous outcomes like mortality. These models, while effective, are highly specific and do not generalize easily to other outcomes.

This research describes the use of non-parametric unsupervised learning to discover abstract patient states that summarize a patient's physiology. The resulting model focuses on grouping physiologically similar patients instead of predicting particular outcomes.

This type of cluster analysis has traditionally been done in small, low-dimensional, error-free datasets. Since our real-world clinical dataset affords none of these luxuries, we describe the engineering required to perform the analysis on a large, high-dimensional, sparse, noisy and mixed dataset.

The discovered groups showed cohesiveness, isolation and correspondence to natural groupings. These groups were also tested for enrichment towards survival, Glasgow Coma Scale values and critical heart rate events. In each case, we found groups which were enriched and depleted towards those outcomes.

Thesis Supervisor: Prof. Peter Szolovits
Title: Professor of Computer Science and Engineering

Thesis Supervisor: Dr. Rohit Joshi
Title: Postdoctoral Associate

Acknowledgments

Even though the thesis describes models based on unsupervised learning, I am extremely fortunate to have undergone supervised learning at the hands of Prof. Peter Szolovits and Dr Rohit Joshi. I am grateful to Prof. Szolovits for his patience in explaining difficult concepts, insight into modelling and guidance throughout the process. I remain in awe of the breadth and depth of his knowledge of machine learning. I am similarly grateful to Dr. Rohit Joshi for his valuable advice, insight into modeling and mentorship.

My colleagues Arya Tafvizi, Marzyeh Ghassemi, Jeremy Lai, Burkay Gur, Skye Wanderman-Milne, Eren Sila Sayan, Andreea Bodnari, Dr. Bill Long, Fern DeOliveira, and others in the Clinical Decision Making group provided wonderful company and made MEDG a very pleasant place to work in.

Prof. Hari Balakrishnan, and Prof. Jacob White graciously supported me via a TAship for 6.02 in Fall 2010. The TAship gave me an opportunity to work with wonderful people like Prof. George Verghese, Prof. Alexandre Megretski, Tanya Khanna, Grace Woo, Xue Feng, Ben Moss, Eric Winokur and Katrina LaCurts.

The past 5 years at MIT have been an exhilarating experience and I am grateful to have had this opportunity. I would like to thank the professors in the EECS department for sharing their wisdom and their contagious enthusiasm. In particular, I am grateful to have had the opportunity to study and urop with Prof. Gerry Sussman whose books and classes have shown me the joy in programming and greatly affected the way I think.

This thesis made extensive use of Free Software such as GNU/Linux, Screen, Emacs, Git, R, AucTex, Org Mode and LaTeX. I would like to thank the contributors of these projects for making their works free for users to use and modify.

Finally, none of this would be possible without the unbounded love and support of my family.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The creation of medical models used to be hobbled by the dearth of data to work with. Since data had to be collected specifically for each task, data collection was an expensive and time-consuming affair. As a result, the key challenge for researchers was to create medical models based on small datasets that were still statistically significant and had statistical power.

In the past decades, there has been an explosive growth in amount collected passively by monitors in hospitals. In particular, bedside monitors in the Intensive Care Units of hospitals now record a gamut of information ranging from blood pressure to lab results on a continuous basis. As a result, we now have a wealth of information on tens of thousands of patients. The development of models using these datasets is an attractive proposition since the cost of data acquisition is effectively nil.

Unfortunately, this type of "ambient" data is frequently recorded directly from the monitoring devices with minimal human intervention, and is therefore noisier than hand-collected data. Thus, medical decision makers are now faced with the problem of reasoning with a dataset that is large, high-dimensional, and noisy. Mechanisms that could reduce the size of the dataset in a noise-resilient manner, while retaining the salient features of the data would go a long way in aiding both reasoning and creation of medical models on these datasets.

This thesis discusses the use of non-parametric unsupervised learning to discover abstract patient states to summarize the condition of a patient. We try to capture the entire physiological state of all organ systems of a patient at a single point in time by analyzing parameters pertaining to the entire body. Since this type of learning has traditionally been limited to smaller error-free datasets, we also describe the steps taken to perform the learning in a real-world medical dataset. We then discuss the validation of the discovered patient states by checking for three outcomes: survival, Glasgow Coma Scale values, and Critical Heart Rate Events.

1.2 Related Work

Previous work with such "ambient" medical datasets has focused almost exclusively on the use of supervised models that predict the onset of hazardous outcomes. For example,

- Hug [23] describes the creation of a model that can predict the trajectory and survival probability of patients in Intensive Care Unit
- Hug [24] expands the work in Hug [23] and describes models to predict the onset of outcomes such as Mortality, Septic Shock and Kidney Injury
- Celi [8] describes a model for mortality that performs significantly better on patients with Acute Kidney Injury than the traditional SAPS score

While these models have been spectacularly successful, they are highly specific to the task and do not generalize. For example, the models trained for Septic Shock prediction are specific to Septic Shock and do not perform as well when predicting a different outcome such as Mortality. As a result, the gains in dimensionality reduction of the dataset using these types of models are limited to specific tasks.

This thesis explores the use of techniques from unsupervised learning to "compress" the dataset by discovering abstract patient states that summarize the condition of a patient in Intensive Care Unit. In other words, we do not train a model that is tuned to predict any specific outcome such as mortality. Instead, we let the data speak for itself and try to find groups of patients who are physiologically similar. The hope is that this compressed dataset could then be used as a basis to create other models, such as a model for the temporal evolution of patient state in response to interventions in the Intensive Care Unit.

This type of non-parametric "compression" of data has been successfully performed in a number of fields. In the following sections we describe a few examples from different areas that served as an inspiration for this work.

1.2.1 Clustering in Critically Injured Patients

The direct inspiration for this work comes from Cohen et al. [11] who used hierarchical clustering to group 52,000 minute-by-minute data points with 40 numeric variables of 17 patients in a surgical intensive care unit. Our work differs in several important ways.

Our work is performed on a substantially larger dataset with over 1 million data points corresponding to over 10,000 patients. Our dataset is also more complex in that it has hundreds of variables and is not restricted to continuous and complete features. Thus, part of the contribution of the thesis is the description of the engineering required to perform cluster analysis on a large, high-dimensional, mixed and incomplete dataset.

We are also more principled in our methodology:

- We account for differences in scales and variance of different features by performing normalization

- We use techniques from the clustering literature to both select the number of clusters to discover and to validate the discovered results

- The k clusters discovered by cutting off the dendrogram discovered by hierarchical clustering is generally suboptimal compared to the k clusters discovered by flat clustering [15, 28]. As a result, we use model selection to find the optimal value of k and use partitional clustering to discover the best clustering of size k.

1.2.2 Topic Modeling

The Topic Modeling problem from Natural Language Processing [38, 5] involves grouping documents that cover similar topics. More precisely, the input to the problem is a corpus of n documents, $d_1, \cdots, d_n$ and a parameter describing the number of groups (either a fixed number k, or a hyperparameter that describes the generative process that is assumed to have created these documents). The topic modeling algorithm needs to group the n documents so that the documents in each group are similar to each other. For example, given a set of news articles, the topic modeling algorithm would be expected to group the international news articles in one group, the sports news articles in another and so on.

This problem is very similar to ours in a number of ways. First, it involves the discovery of groups for a set of data points. These groups are intuitive and easy to explain (e.g. documents that discuss the same topic, patients who have similar physiology) but are not as well-defined mathematically (e.g. What is a topic? What does "similarity" in the context of ICU patients mean?).

Second, both problems involve data in large dimensions. Topic Modeling is traditionally done with the *bag-of-words* model where the count of each word (or sequence of words) is considered an independent feature. Thus, the dimensionality of these datasets can easily be in the tens of thousands.

However, popular topic model algorithms generally assume that the documents are generated according to some simple probability distribution, like the Dirichlet distribution. It is not clear whether human physiology can be described using a similarly simple probability distribution. In this thesis, we choose to work in a non-parametric manner and do not assume that the data is generated by some overarching simple probability distribution[1].

[1]The use of model-based clustering methods is briefly discussed in Chapter 5

1.2.3 Vector Quantization

Vector Quantization [22] is a technique for compressing images so that the resulting image looks similar to the original while requiring much less storage.

Images are represented by a two-dimensional array of "pixels", which represent the color at a point. A pixel traditionally requires 32-bits to represent the color: 8-bits each for Red, Blue and Green components and a further 8-bits to represent transparency information. Thus, a typical high-resolution image with 1920-by-1080 pixels would require $1920 \times 1080 \times 32 = 66355200$ bits = 7 megabytes to store.

The idea behind vector quantization is to reduce the size of the image by reducing the number of bits required to describe each pixel. More precisely, the colors are partitioned into K groups (usually a few hundred) so that similar colors are grouped together. Then, each color is replaced by the arithmetic mean of the colors in the group. As a result, we can now store the K "representative" colors in $K * 32$ bits and use $log_2 K$ instead of 32 bits per-pixel to describe the color.

The work in the thesis is similar to Vector Quantization since it also uses non-parametric unsupervised learning techniques to reduce the size of the dataset by grouping similar items together. Just as the quality of image compression is judged by similarity between the compressed image and the original, we judge the quality of our groups by ensuring that the groups found are medically significant and that the compression hasn't led to a loss of the salient features of the dataset.

1.3 Thesis Organization

The thesis is organized as follows:

- Chapter 2 introduces the MIMIC II database and describes the creation of the dataset which we work with. It also describes the problems with the dataset and the techniques applied to ameliorate them.

- Chapter 3 describes clustering, the non-parametric unsupervised learning technique used in this thesis. It introduces the technique and describes the refinements required to use it on our dataset.

- Chapter 4 describes the selection of the clustering model and its validation. The model is validated by checking goodness of clustering with a statistical measure and by checking if the resulting clusters are enriched towards some medically significant outcomes.

- Finally, Chapter 5 summarizes the contributions of the thesis and discusses some avenues for future research.

Chapter 2

The MIMIC Dataset

This chapter describes the creation of a dataset from the Multiparameter Intelligent Monitoring in Intensive Care (MIMIC II) database [37]. The MIMIC II database contains detailed deidentified physiological information on patients in the Intensive Care Units (ICU) of the Beth Israel Deaconess Medical Center collected over a seven year period from 2001. The data contains information on the patients in five adult ICUs (Coronary, Trauma, Surgical, Medical and Cardiac) and the Neonatal ICU. This data was gathered to enable research in patient monitoring techniques and has successfully been used to create models that predict the onset of hazardous events such as mortality, organ failures and septic shock [8, 23, 24].

The MIMIC database is further divided into two sub-databases: the *High-Resolution Waveform Database* and the *Clinical Database.* The Waveform Database consists of data recorded from bedside monitors such as Electrocardiograph Monitors (ECG), Ambulatory Blood Pressure Monitors (ABP) and Respiratory Monitors for over 2000 patients. This data is collected at a frequency of 125 Hertz and has a 8-10 bit resolution.

The Clinical Database consists of time-series data collected from the Philips CareVue monitors and the Hospital Archives for over 32,000 patients. This database is organized around the concept of an *event*, which is a record of an activity such as administration of medication or measurement of vitals along with a timestamp. Thus, the Clinical Database can be viewed as a time-series data of patients that records medications administered, fluid input and output events, nursing notes, and lab results of the patients.

This thesis focuses on performing unsupervised learning in data extracted from the Clinical Database since there is very little overlap in the techniques and concepts between clustering of continuous high-frequency signal data and the clustering of more discrete data. Some suggestions for performing unsupervised learning in signal data are presented in Chapter 5.

2.1 The Clinical Database

The Clinical Database is organized as a relational database that contains information on both patients and their caregivers. The data for each patient is identified by a combination of the following IDs:

- `Subject_ID` which distinguishes between different patients
- `Hadm_ID` which distinguishes between different hospital admissions of the same patient
- `ICUStay_ID` which distinguishes between stays in different ICUs during a hospital admission

The patient data is grouped by type of data and stored in the following tables:

- `Chart` data from a patient's medical chart such as nurse-validated readings, severity scores and ventilator settings
- `Demographics` such as age, sex, ethnicity, height and admission weight.
- `Fluids` administered and withdrawn from patients
- `Laboratory` test data such as blood gas and chemistry
- `Medications` administered intravenously to patients
- `Notes` written by the nurses in free-text format

The data in these tables are organized around the concept of an *event*. For example, when a patient is given Insulin (a medication), a new event is created that contains the patient's ID, the type of medication administered, the time of administration and other details such as dosage and administration method. While the columns used to record extra detail varies by table, each table has the following information:

- `Subject_ID`, `Hadm_ID`, and `ICUStay_ID` to identify the patient
- `ItemID` of the medication, laboratory test, fluid input output type, demographic detail or chart item
- `ChartTime` the time at which the event occurred
- `Value` of lab result, dose of medication, amount of fluid input output.

The different items in each table are distinguished by `ItemID`s. For example, Insulin in the medications table has an `ItemID` of 45 while Integrelin has an `ItemID` of 142. Unfortunately, there are a few dozen features that have multiple `ItemID`s (for example, the medication Neosynephrine has `ItemID` 127 as well as 128). We used the table in [10] to identify these features and took care to combine their observations.

2.2 Feature Selection

Every `ItemID` in the database corresponds to a feature that can be used to build a model. In order to understand the dimensionality, we counted the number of unique `ItemID`s in the database. These counts are presented in Table 2.1. The table shows that the database has roughly 13,000 features and is thus very high dimensional.

In order to understand the sparsity of the data, we counted the number of patients who had a value for each `ItemID`. We found that the counts followed a power-law distribution with a few features being present in a significant number of patients and most features restricted to a handful of patients.

This combination of high dimensionality and sparsity is problematic for several reasons. First, as we will see in Chapter 3, the time taken to cluster a dataset increases linearly with the number of features. So, the presence of a large number of features increases the time to create a model. Second, missing values in a feature have to be accounted for somehow when comparing two data points to understand their similarity[1].

Thus, we applied some heuristics to reduce the number of features considered by the model. First, we retained each of the roughly 400 features used by [23, 24] in his models to predict hazardous events because they showed good predictive power towards medically significant outcomes. We then supplemented this feature set by adding a handful of features discussed in the medical literature as being medically relevant. Finally, we retained any feature that was present in a significant number of patients. For example, the cutoff for the Medications table was 1000 patients. The selected features are listed in Tables 2.2 – 2.8. The columns for "Type" and "Hold Limit" in these tables are explained in Chapter 3.

These features were augmented by incorporating two types of derived variables as was done in [24]. First, we computed the *slope* of a feature over a course of time (usually 240, 1440 or 1680 minutes). Second, we counted the number of minutes a variable was out of normal range. These variables allow us to capture evolutionary trends in the feature, and allow us to incorporate some of the information from the patient history. A more principled approach to incorporating time series information would be to use techniques from non-parametric time series clustering such as Dynamic Time Warping[2].

2.3 Data Reshaping

Clustering algorithms require the input data to be represented by a $n \times f$ matrix where the n rows are the observations and the f columns are the features of the dataset. Thus, we had to extract the data from the relational form into this format

[1]Missing variables refers to measurements which are absent, such as the absence of fluid input output measurement at a certain time. It does not, for example, refer to the lack of administration of a certain medication; these are simply marked as False.

[2]This and other techniques are suggested in Section 5.2.

Table	Number of features
Chart	4832
Demographics	88
Fluids	6808
Laboratory	713
Medications	405
Total	12846

Table 2.1: The number of features available in the MIMIC II Clinical Database. This table illustrates the high dimensionality of the dataset.

to develop our models.

The extraction was done in two steps. First, we transformed each table so that the selected features were now the columns of the table. For example, the transformed Medications table had the following columns: Subject_ID, Chart Time, Integrelin, Ephinephrine, Lasix, Vasopressin, Nitroprusside, and so on. That is, each type of medication is now a column in the table whose values are True if the medication was administered and False otherwise.

We then performed a *table join operation* on the transformed tables to obtain a single table that contained all the columns. The rows for the same `Subject_ID` from different tables were combined using the following strategy:

- Rows for the same `ChartTime` were merged into a single row
- Rows for different `ChartTime` were not merged; instead, NULL values were placed in the other columns. For example, if we have Medication data for a certain patient at a certain time but no data on Fluids, all values in the fluid column would be NULL.

2.4 Problems with the Dataset

We will now discuss problems with the extracted dataset and discuss some measures taken to mitigate these problems. Further measures to ameliorate these problems are discussed in Chapter 3.

Feature	ItemID	Type	Hold Limit (Hours)
Integrelin	142	Binary (Asymmetric)	—
Epinephrine	119, 44	Binary (Asymmetric)	—
Lasix	123	Binary (Asymmetric)	—
Vasopressin	51	Binary (Asymmetric)	—
Nitroprusside	50	Binary (Asymmetric)	—
MorphineSulfate	126	Binary (Asymmetric)	—
Amiodarone	112	Binary (Asymmetric)	—
Midazolam	124	Binary (Asymmetric)	—
Dopamine	43	Binary (Asymmetric)	—
Fentanyl	118, 149	Binary (Asymmetric)	—
Levophed	120, 47	Binary (Asymmetric)	—
Heparin	25	Binary (Asymmetric)	—
Nitroglycerine	121, 49	Binary (Asymmetric)	—
Insulin	45	Binary (Asymmetric)	—
Neosynephrine	127,128	Binary (Asymmetric)	—
Propofol	131	Binary (Asymmetric)	—

Table 2.2: Features selected from the Medications table. The "Type" column is explained in Section 3.2.3. The "Hold Limit" column is explained in Section 2.4.2.

Feature	ItemIDs	Type	Hold Limit (hours)
Systolic Blood Pressure	455	Numeric	4
Diastolic Blood Pressure	455	Numeric	4
Mean Arterial Pressure	456	Numeric	4
NBP	1149	Numeric	4
Arterial Blood Pressure (Systolic)	51	Numeric	4
Arterial Blood Pressure (Diastolic)	51	Numeric	4
Arterial Blood Pressure (Mean)	52	Numeric	4
Heart Rate	211	Numeric	4
Oxygen Saturation	646, 1148	Numeric	4
Central Venous Pressure	113, 1103	Numeric	4
PAP Mean	491	Numeric	4
PAP Standard Deviation	492	Numeric	4
Cardiac Index	116	Numeric	4
SVR	626	Numeric	4
CO (thermodilution)	90	Numeric	4
CO (fick)	89	Numeric	4
PCWP	504	Numeric	4
PVR	512	Numeric	4
Cardiac Murmur	3353	Numeric	4
Vitamin K	3685	Numeric	4

Table 2.3: Cardiovascular features selected from the Charts Table

Feature	ItemIDs	Type	Hold Limit (Hours)
Sodium	837, 1536	Numeric	28
Potassium	829, 1535	Numeric	28
Chlorine	788, 1523	Numeric	28
Phosphorous	827	Numeric	28
Lactic Acid	818, 1531	Numeric	28
Carbon Dioxide	787	Numeric	28
Glucose	811	Numeric	28
Blood Urea Nitrogen	781, 1162	Numeric	28
Creatinine	791, 1525	Numeric	28
Magnesium	821, 1532	Numeric	28
Calcium	786, 1522	Numeric	28
Ionized Calcium	816	Numeric	28
ALT	769	Numeric	28
AST	770	Numeric	28
Troponin	851	Numeric	28
Fibrinogen	806	Numeric	28
Total Bili	848, 1538	Numeric	28
Direct Bili	803, 1527	Numeric	28
Total Protein	849, 1539	Numeric	28
Albumin	772, 1521	Numeric	28
Lactic Acid	818, 1531	Numeric	28

Table 2.4: Chemistry features selected from the Charts Table

Feature	ItemIDs	Type	Hold Limit (Hours)
Arterial Base Excess	776	Numeric	28
Arterial CO2	777	Numeric	28
Arterial PaCO2	778	Numeric	28
Arterial PaO2	779	Numeric	28
Arterial pH	780, 1126	Numeric	28
Venous PvO2	859	Numeric	28

Table 2.5: Blood Gas features selected from the Charts Table

Feature	ItemIDs	Type	Hold Limit (Hours)
FiO2 Set	190	Numeric	28
PEEP Set	506	Numeric	28
Respiration Rate	618	Numeric	28
Respiration Rate Total	615	Numeric	28
Respiration Rate Set	619	Numeric	28
Respiration Rate Spon	614	Numeric	28
Peak Insp. Pressure	535	Numeric	28
Plateau Pressure	543	Numeric	28
Tidal Volume (Observed)	682	Numeric	28
Tidal Volume (Set)	683	Numeric	28
Tidal Volume (Spont)	684	Numeric	28
SaO2	834	Numeric	28
Lung Sounds	428, 425	Nominal	28

Table 2.6: Ventilation features selected from the Charts Table

Feature	ItemIDs	Type	Hold Limit (Hours)
Hematocrit	813	Numeric	28
Hemoglobin	814	Numeric	28
INR	815, 1530	Numeric	28
Platelets	828	Numeric	28
PT	824, 1286	Numeric	28
PTT	825, 1533	Numeric	28
White Blood Cells	861, 1127, 1542	Numeric	28
Red Blood Cells	833	Numeric	28
Temperature	678, 679	Numeric	28

Table 2.7: Hematology features selected from the Charts Table

Feature	ItemIDs	Type	Hold Limit (Hours)
Heart Rhythm	212	Ordinal (Symmetric)	3
Ectopy Type	161	Ordinal (Symmetric)	3
Ectopy Frequency	159	Ordinal (Symmetric)	3
Code Status	128	Ordinal (Symmetric)	3
Risk for Falls	1484	Ordinal (Symmetric)	3
Orientation	479	Ordinal (Symmetric)	3
Level of Consciousness	432	Ordinal (Symmetric)	3
Eye Opening	184	Ordinal (Symmetric)	3
Motor Response	454	Ordinal (Symmetric)	3
Riker SAS	1337	Ordinal (Symmetric)	3
Ventilator Type	722	Ordinal (Symmetric)	3
Ventilator Mode	720	Ordinal (Symmetric)	3
Pacemaker Type	516	Ordinal (Symmetric)	3
Trachea Size	690	Ordinal (Symmetric)	3
Skin Color	643	Ordinal (Symmetric)	3
Skin Integrity	644	Ordinal (Symmetric)	3
Service Type	1125	Ordinal (Symmetric)	3

Table 2.8: Categorical features selected from the Charts Table

2.4.1 Large Size

The dataset as we've extracted contains over 10,000 patients and 1 million rows. We followed the lead of [24] and performed several operations to restrict the patients we considered.

First, we considered only those patients who were admitted into the adult intensive care units since the physiology of neonates is markedly different from the physiology of adults. The comparatively smaller number of data for neonates would mean that these would simply manifest themselves as outliers.

Second, we removed patients who were discharged within a few hours of admission into the ICU, or did not have information for the following very common variables:

- Blood Urea Nitrogen
- Hematocrit
- Heart Rate
- At least one Medication Event

These measures helped remove spurious patients from the dataset. However, we still have a large number of rows due to the way the rows were merged when the table join was performed. We only merged rows which corresponded to exactly the same `ChartTime`. Since the `ChartTime` has a granularity of seconds, only a few rows were actually merged. We decided to change the granularity of the records to 30 minutes[3], and merged rows in 30 minute windows using the following strategy:

- If there was no reading in any of the rows for a particular feature, we set the value as NULL
- If there was only one row with a value for a particular feature, we used that value
- If we had more than one value for the same feature, we used the extreme values. This meant taking the largest value for numeric values except for features measuring the low values such as Diastolic Blood Pressure. The use of extreme values increases our susceptibility to noise and outliers, but these extreme values could indicate problems which would be hidden if we used a central value such as median.

2.4.2 Missing and Irregularly Sampled Data

The frequency at which the data is collected is dependent on a variety of factors such as the monitor type, medical personnel availability and the patient's condition. For example, a patient in critical condition has more frequent lab results than someone in stable condition. This means that the data is not *Missing Completely at Random* and thus, traditional strategies to deal with missing data such as interpolation or imputation will not yield good results unless they explicitly model the reasons why the data is missing [10].

[3]This is an arbitrary choice, and other values such as 60 minutes or even 10 minutes would be valid. We wanted to strike a balance between reducing the number of rows in the table and retaining information in the dataset

We cannot simply ignore the missing data either, since the absence of a value for a particular test could indicate that the physician did not think that the test would give insight into the patient's condition. In other words, the absence of a value for a feature could influence the model by incorporating the physician's understanding of the patient's condition. We tried to minimize this creep-in by discarding the very rare features. For example, we ignored medications that were given to fewer than 1000 patients.

We also used the sample-and-hold technique used by [24] to reduce the number of missing data. The idea behind sample-and-hold is to assume that measurements are valid for a certain period of time, which allows us to replace missing values from past measurements. The period for which a measurement is considered valid was determined empirically by determining the frequency at which the measurement was performed. For example, we found that the patient's Glucose chemistry was measured roughly every 24 hours. So, we treated a measurement of Glucose as valid for the next 28 hours. The hold values used for the variables are listed in Tables 2.2 – 2.8.

Sample-and-hold is a form of simple-minded imputation where we assume that the variable stays constant during the period we don't have the data for and that the variable moves in a step-wise fashion. The use of more sophisticated models of physiological variable evolution could lead to better results.

2.4.3 Bias and Noise

While the MIMIC developers have made a considerable effort to improve the quality of the dataset with each release, the dataset still has noise due to causes ranging from improper instrument calibration to incorrect data entry.

This noise could appear in the dataset as *outliers* (for example, when a temperature is stored as 30 degrees Fahrenheit instead of Celsius), or as small changes in magnitude called *bias*. Suppose a temperature monitor is incorrectly calibrated and always adds 1 degree to the actual temperature. The added noise is small enough that the data points are not detected as outliers, but still represent incorrect measurements.

We perform two operations to minimize the effects of outliers. First, we removed observations which are more than 3 standard deviations away from the observed mean. This three-sigma rule is well-accepted statistical technique of removing outliers in the data [7].

Second, we use robust measures whenever possible. For example, we use median as the measure of central tendency instead of mean when finding representative objects during clustering, and mean absolute deviation instead of sum of squares deviation to measure dispersion. This is discussed in greater detail in Chapter 3.

2.4.4 Non-Numeric Features

Another problem is that the dataset is not entirely numerical. For example, Table 2.8 lists several ordinal variables like Ectopy Frequency, which takes on the values "Rare", "Occasional", "Frequent". These variables are not ratio-scaled so the difference between "Rare" and "Occasional" is not the same as the difference between "Occasional" and "Frequent". Thus, a naive mapping of these features into the numeric scale would lead to incorrect results.

We deal with this mixture of features by developing a metric that can handle mixed data in Chapter 3.

2.5 Summary

This chapter described the MIMIC II database, the source of the data used in this thesis. We described the organization of the Clinical database and the notion of events. We noted the high dimensionality of the dataset and discussed the techniques used to reduce it. We then discussed other problems with the dataset like its large size, missing data, irregularly sampled data, noise, presence of non-numeric measures and described some measures taken to mitigate their effects.

Chapter 3

Clustering

3.1 Introduction

Clustering is an unsupervised learning problem where we are given unlabeled data points and we need to group the data so that data points that are similar to each other are placed in the same group.

For example, consider the "Old Faithful" dataset [2] shown in Figure 3-1. The data describes the time in minutes to the next eruption (y-axis) and the eruption duration in minutes (x-axis) of the Old Faithful geyser in Yellowstone National Park.

We see two distinct groups of data; one located in the bottom-left of the plot, and another in the top-right of the plot. Figure 3-2 shows the result of *k-means*, a clustering algorithm on this dataset, when asked to find 2 clusters in the data. The discovered clusters are plotted with different symbols.

3.1.1 The K-means algorithm

We now discuss the *k-means* algorithm in order to understand the way clustering algorithms work.

The idea behind k-means is to identify k centroids in the dataset and assign each data point to the nearest centroid. The K-means algorithm tries to find centroids and assignments that minimize the sum of squares of the distances between the each

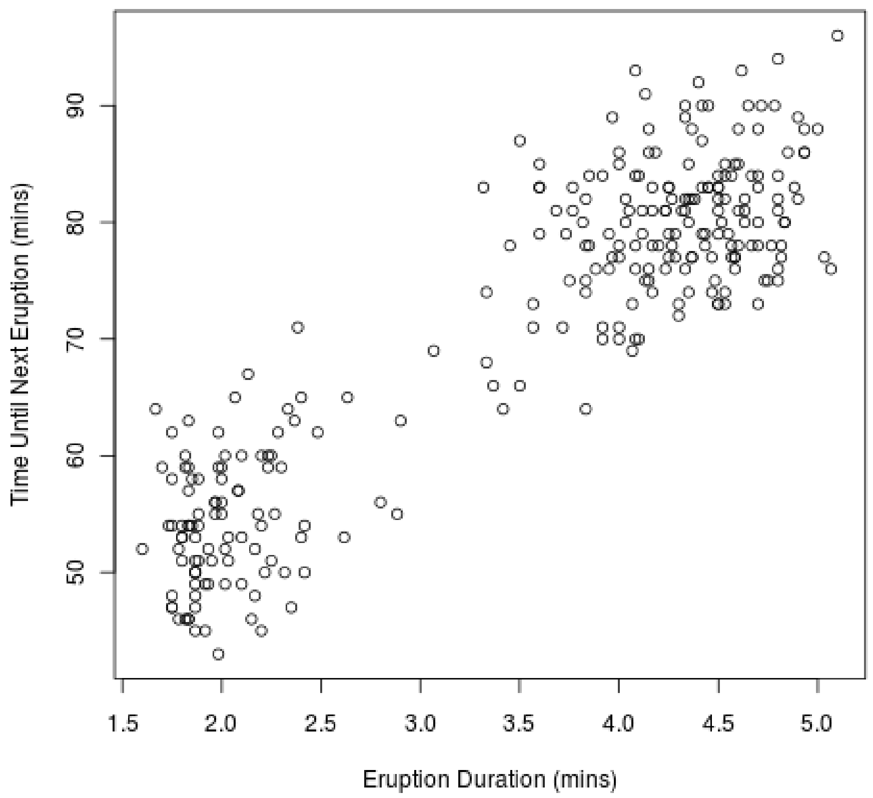

Figure 3-1: Old Faithful data

centroid and the points assigned to it.

Formally, the input of the k-means algorithm are the n data points $x_1, \cdots, x_n \in X$, distance function $d : X \times X \to \mathbb{R}$ and the number of desired clusters k. The output is the set S of k-centroids $\mu_1, \cdots \mu_k$ and clusters $s_1, \cdots, s_k$ such that:

$$S = \arg\min_{s} \sum_{i=1}^{k} \sum_{x_j \in s_j} (d(x_j, \mu_i))^2 \tag{3.1}$$

and

$$X = \bigcup_{c=1}^{k} (x \in s_c) \tag{3.2}$$

Since this optimization problem is NP-hard in a general Euclidean space, it is traditionally solved using an iterative approximation algorithm[1]. The algorithm is

[1]This algorithm is very similar to the Expectation Maximization algorithm [13] used to estimate

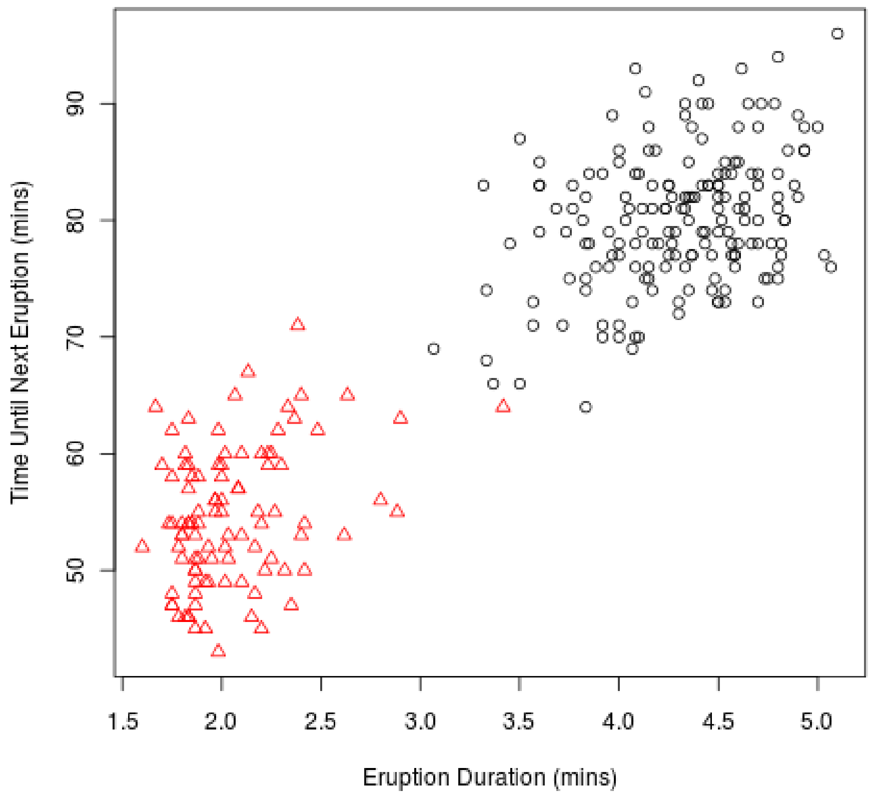

Figure 3-2: Clusters in the Old Faithful data

initialized with k random points, which are the initial guesses of the centroids. It then improves the guess for centroids by alternating between two steps: *assignment* and *update*.

In the assignment step, each data point is assigned to the closest centroid. In the update step, new centroids are computed using all the data points that were assigned to the same centroid. These two steps are repeated either for a fixed number of iterations or until convergence.

However, this algorithm can get stuck in a local optima, so it is necessary to run the algorithms several times with different starting points.

the parameters in statistical models.

3.1.2 Structure of the chapter

We can identify three critical components of a clustering algorithm from the description of the k-means algorithm:

- The distance metric $d(x_i, x_j)$ used to quantify the distance between two points
- The number of clusters to be found (k)
- The optimization criteria and the algorithm to solve the optimization problem

We will now discuss each of these components in turn.

3.2 Distance Metrics

We begin by formalizing the notion of dissimilarity and describing the dissimilarity metrics for homogeneous data. We will then enhance these metrics to allow them to deal with heterogeneous features, features in different scales, and features with missing data.

3.2.1 Quantifying Dissimilarity

The notion of dissimilarity or distance between two data points is captured by the notion of *metric* from topology [36, 32]. A metric on a set X is a function $d : X \times X \to \mathbb{R}$ that satisfies the following conditions:

- **Symmetry** $d(x, y) = d(y, x)$
- **Non-negativity** $d(x, y) \geq 0$, with equality when $x = y$
- **Triangle Inequality** $d(x, z) \leq d(x, y) + d(x, z)$

The use of distance functions that are not valid metrics to perform clustering yields meaningless results.

3.2.2 Metrics for Numeric Features

We start by assuming that the dataset contains p features, all of which are numeric. The data points will be denoted by $x_1, \cdots, x_n$ and the value of the f^{th} feature in the j^{th} data point will be denoted by $x_{i,f}$.

Metrics used for numeric data can be broadly categorized into those that are based on distance and those that are based on correlation [15]. We will focus on distance-based metrics in this thesis because of their widespread-use and conceptual simplicity.

A commonly used numeric metric is the Euclidean distance[2]:

$$d(x_i, x_j) = \sqrt{\sum_{f=1}^{p} (x_{i,f} - x_{j,f})^2} \tag{3.4}$$

The Euclidean distance can be interpreted as the distance between two points in a p-dimensional Euclidean space.

3.2.3 Metrics for Nominal Features

Nominal features take on a finite number of unordered values. For example, the feature "Sex", which in the MIMIC dataset has two possible values "Male" and "Female", is a nominal variable. This feature is unordered because we cannot say that "Male" is greater than "Female" or vice-versa.

In order to compute a distance between two data points, x and y whose features are all nominal, we use metrics derived from *contingency tables* [7]. For simplicity, we assume that each feature can only take on two possible values, arbitrarily labeled 0 and 1. The contingency table for x and y is shown in Table 3.1.

We obtain metrics from the contingency table by combining the counts of the times x and y had similar values (a and d) with the count of times they had different

[2]This metric is a specific instance of the Minkowski-distance with $k = 2$:

$$d(x_i, x_j) = (\sum_{f=1}^{p} |x_{i,f} - x_{j,f}|^k)^{\frac{1}{k}} \tag{3.3}$$

	y = 1	y = 0
x = 1	a	b
x = 0	c	d

Table 3.1: A contingency table for two data points x and y. Each cell contains the count of features where x and y had the specified value. For example, "a" is the number of features where both x and y are 1.

values (b and d). The *Simple Matching Coefficient* [28, 41] is one such metric:

$$d(x, y) = \frac{b + c}{a + b + c + d} \tag{3.5}$$

It returns the percent of features where the two data points had different values. It also has the desirable property that it returns the same value even when some or all of the labels are swapped.

There is a slight complication when we have features which have outcomes of different weights. For example, the feature "AIDS" has values "False" and "True" indicating the absence (and presence) of AIDS in the patient. How do we encode our expectation that two patients who have AIDS are more similar than two patients who do not have AIDS?

One way to do it is to assign weights to the different outcomes. Jaccard's Coefficient [28, 25] deals with this asymmetry in outcomes by assigning a 0 weight to the less-informative outcome[3]:

$$d(x, y) = \frac{b + c}{a + b + c} \tag{3.6}$$

Comparing this to 3.5, we see that the value d which corresponds to the $x = 0, y = 0$ outcome has been omitted.

So, we can compute distance between two points in a dataset that contains only nominal variables using the Simple Matching Coefficient when the features have equally informative outcomes and Jaccard's Coefficient when the features have outcomes that are not equally informative.

[3]A more sophisticated approach is to use real-valued weights (obtained using techniques like regression) instead of just ignoring uninformative outcomes.

3.2.4 Dealing with Data in Different Scales

The MIMIC dataset contains information pertaining to a wide variety of medical measurements. As a result, the numeric values reflected in different columns are in different scales. For example, the "Temp" column, for patient temperature, ranges between 80 and 108 degrees Fahrenheit, while the "AmiodaronePerKG" column ranges between 0 and 3.5. We need to prevent the larger values of temperature from overwhelming the smaller values from AmiodaronePerKG when we compute the distance.

Besides a difference in units, we also find a difference in the variance of different columns. For example, even though AmiodaronePerKG and AmicarPerKG have the value that are in the same range and unit, values in AmiodaronePerKG are more dispersed. This means that the same magnitude of change in is more significant in AmicarPerKG than in AmiodaronePerKG.

The differences in scale and dispersion are typically dealt with by standardizing the values so that each feature has 0 mean and unit standard deviation. The result values are called *Z-scores.*

More precisely, suppose $x_1, x_2, \cdots x_n \in X$ are the data points, f is a particular feature (for example, temperature) whose mean is μ_f and standard deviation is σ_f. Let $x_{i,f}$ denote the value of feature f in the i^{th} data point, and $z_{i,f}$ denote the z-score of feature f for the i^{th} data point. Then,

$$z_{i,f} = \frac{x_{i,f} - \mu_f}{\sigma_f} \tag{3.7}$$

A problem with the Z-Score is that it relies on mean and standard deviation, which are not robust [7, 28]. The lack of robustness is problematic because our dataset is noisy. We follow the advice of [28] and use the more robust *Mean Absolute Deviation* measure [19] instead of standard deviation. The Mean Absolute Deviation (mad_f) of a feature f is defined as follows:

$$mad_f = \frac{\sum_{i=1}^{n} |x_{i,f} - \mu_{i,f}|}{n} \tag{3.8}$$

Thus, we deal with the difference in scales among different features by normalizing to the following robust variant of the Z-score:

$$z'_{i,f} = \frac{x_{i,f} - \mu_f}{mad_f} \tag{3.9}$$

3.2.5 Dealing with Mixed Data

We now relax the assumption that the dataset is composed of homogeneous features since it is inconsistent with our dataset. We discuss the construction of a new metric that works with heterogeneous features.

Suppose we group our features into numeric, symmetric nominal, asymmetric nominal and compute distances for each type using Euclidean distance, Simple Matching Coefficient and Jaccard's Coefficient respectively.

Each of these distances are then rescaled to lie between 0 and f_t where f_t is the number of pairwise-complete features considered when computing the distance of the type t[4]. This ensures that the final distance has a contribution proportional to the number of features from each type of data. We can now define a new metric, similar to the coefficient developed by Gower [18], that is simply a linear combination of these metrics:

$$d(x_i, x_j) = D_{numeric}(x_i, x_j) + D_{symmetric}(x_i, x_j) + D_{asymmetric}(x_i, x_j) \tag{3.10}$$

where $D_{numeric}(x_i, x_j)$ is the scaled Euclidean distance between all the numeric features of x_i and x_j rescaled to lie between 0 and 1 ($D_{symmetric}$ and $D_{asymmetric}$ are similarly defined). This function satisfies the requirements of a metric function mentioned in Section 3.2.1.

[4]The Simple Matching Coefficient and Jaccard's Coefficient are already normalized since the score for each feature lies between 0 and 1.

3.2.6 Dealing with Missing Data

As mentioned in Section 2.4.2, missing data is pervasive in “found” datasets such as MIMIC. This is true even after using techniques like holding and getting rid of extremely rare features (discussed in Section 2.4.2). In denser datasets, we could remove all the data points with missing data and still be left with a sizable dataset, but the MIMIC data is sparse enough that this type of removal would result in an empty dataset.

A slightly more sophisticated solution would be to remove all the “rare” features in the dataset before removing data points with missing values. This approach doesn't work well either; if we were to keep only the features which were present in over 90% of the data points, we would end up losing around 150 of the 450 features. Following this by removing data points with any missing value removes 25% of the available data.

Since it is unacceptable to lose so many columns and data points, we use the strategy proposed by [28]. First, we compute the per-feature distance (as described in 3.2.5) for each *pairwise-complete feature*, that is, features for which we have values for both x_i and x_j. If q out of the p features are pairwise complete, we weigh compute the sum of the distance between each feature and multiply it by $\frac{q}{p}$. For example, the formula for euclidean distance with missing data is:

$$d(x_i, x_j) = \sqrt{\frac{p}{q}((x_{i,1} - x_{j,1})^2 + \cdots + (x_{i,p} - x_{j,p})^2)} \tag{3.11}$$

This formula blows up when two data points don't have any pairwise-complete features ($q = 0$). However, in this case we don't have any common measurements between two data points, so the data points are incomparable. To prevent problems of this type, we remove pairs of data points which don't have pairwise-complete features in a greedy manner starting with data points which have the largest number of missing features.

It is important to note that this metric becomes increasingly inaccurate as q increases. As such, it is not a silver bullet to the problem of missing data and should be used in conjunction with more sophisticated techniques such as holding and imputation.

3.3 Number of Clusters in a Dataset

K-means is a partitional clustering algorithm; it partitions the dataset into k clusters where k is specified by the user. How do we determine the right value of k to use?

In the Old Faithful clustering example (3.1), we simply eyeballed the data plot and found 2 natural clusters. There are several problems with this eyeballing technique. First, it is ill-defined because it relies on subjective groupings discovered by humans; even reasonable people can disagree on the number of groups when the data is less obviously demarcated.

Second, our intuitive notion of distance between two points in a plane is the Euclidean distance. However, things that are close according to the Euclidean distance need not be close according to a different metric. Thus, the eyeballing technique would force us to use the Euclidean distance while clustering.

Third, it relies on the visualization of data points in the feature space. It is easy to visualize points in a 2-dimensional space as in the Old Faithful data, but much harder to do the same in a dataset in hundreds of dimensions.

Clearly, we need a more principled method to find the number of clusters in a dataset. One way to do this is to try different values of k and compare the quality of clusters that are produced[5]. Unfortunately, the indices to measure cluster quality are a dime a dozen [28, 40, 31, 17] and the literature provides no guidance on the appropriate index.

I used one index to determine the value of k during model selection, and then validated the choice by using a different metric. I will now describe the index used during construction (the other indices will be described as they are introduced in

[5]Another way is to assume that the data is generated by a generative process and use an infinite relational model such as the Chinese Restaurant Process [4]. Instead of taking a fixed number k as input, these models take in parameters that control the creation of new clusters/groups. Thus, these methods can use "as many clusters as necessary." One of the problems is that these techniques are very inefficient compared to the more pedestrian methods.

Chapter 4).

3.3.1 Silhouettes

I decided to use the *Silhouette Coefficient* [34, 28] during model construction based on the recommendation of [28]. The Silhouette Coefficient of a clustering is derived from the *Silhouette Width* of points in each cluster. Silhouette Widths compare the average distance between a point and other points in its cluster with the average distance between the point and other points in the next nearest cluster. This allows us to identify situations where we have points that could just as well be in a different cluster (which is typically the case when k is both too small or too large).

Mathematically, the Silhouette Width $s(i)$ of a point i is defined as follows. Let $a(i)$ be the average dissimilarity between the point i and all other points in the same cluster as i. Let $d(i, C)$ be the average dissimilarity between i and points in a cluster C, where $C \neq A$. Then, $b(i) = min_{C \neq A} d(i, C)$ represents the average dissimilarity between i and the next nearest cluster to i. Then, $s(i)$ is:

$$s(i) = \frac{b(i) - a(i)}{max(a(i), b(i))} \tag{3.12}$$

So, the silhouette width of a point is a number between -1 and 1. Positive values of the width imply that the within cluster dissimilarity is smaller than the dissimilarity with the next best index, suggesting good clustering. Negative values suggest that the clustering could be improved by placing this point in a different cluster.

We define the Silhouette Width of a cluster to be the average of the silhouette widths of each point in the cluster, and we define the Silhouette Width of a clustering assignment to be the average of the Silhouette Widths of each cluster. We prefer clustering assignments with higher Silhouette Widths. Thus, we can select the best value of k using:

$$k^* = \arg\max_k s_{clustering}(k) \tag{3.13}$$

3.3.2 Example: Silhouette Coefficients with Old-Faithful data

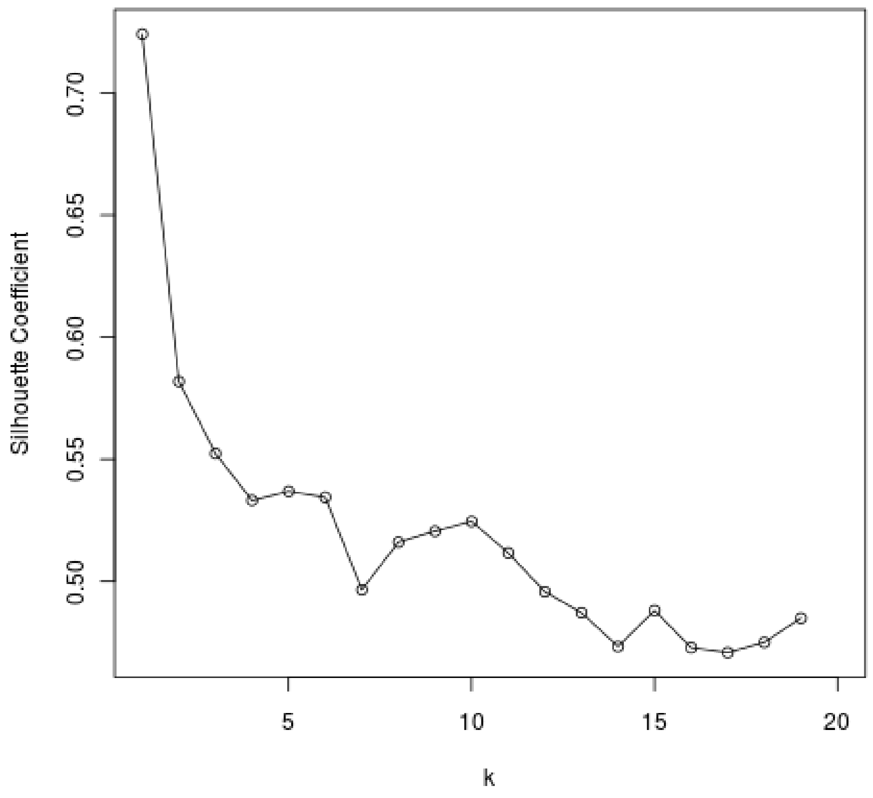

Figure 3-3: Silhouette Coefficients for various values of k in the Old Faithful dataset. Silhouette Coefficient values range from +1 to -1 with large values indicating good clustering. We see the maximum value at $k = 2$ which validates our intuition that the dataset has two clusters.

Figure 3-3 shows the values of silhouette coefficients for different values of k when clustering the Old Faithful dataset using k-means with Euclidean distance. The best value, 0.7241, is obtained at $k = 2$, which is what we expected based on eyeballing the data. Furthermore, according to [28], silhouette values higher than 0.70 indicate the discovery of a strong structure, so we can be reasonably confident in our choice of k.

3.4 The Clustering Algorithm

We now discuss the clustering algorithm used in the thesis. We begin by discussing some limitations of the simple k-means algorithm discussed in Section 3.1.1. This discussion leads us to the *Partitioning Around Medoids* algorithm [28]. We then describe the time and space efficiency of clustering algorithms based on the *distance-matrix approach* such as k-means and partitioning around medoids. This discussion leads us to *Clustering Large Applications* algorithm [28] which is an approximation of partitioning around medoids.

We motivate the development of the *Partitioning Around Medoids* algorithm [28] by discussing two problems with the k-means algorithm. First, k-means uses centroids (mean values) instead of medoids (median values). This is problematic because the mean is more sensitive to outliers, and hence less robust than median [7, 19]. Furthermore, unlike the median, the mean of a set of points need not be a point in the set. This means that we cannot interpret the discovered cluster centers as "representative elements" of the clusters. Thus, the partitioning around medoids algorithm uses medoids instead of centroids.

Second, the k-means algorithm tries to minimize sum of square of distance between a point and its cluster center. As in Section 3.2.4, we note that methods based on minimization of sums of squares are less robust than methods that minimize sums or averages [7, 28, 19]. For this reason, the optimization criterion of the partitioning around medoids algorithm minimizes average distance to cluster center.

3.4.1 The Partitioning Around Medoids Algorithm

We now present the *Partitioning Around Medoids* [28, 26] algorithm that is the foundation of the clustering algorithm used in the thesis. The algorithm works in two stages, called `BUILD` and `SWAP`. We will discuss each stage in turn.

During the `BUILD` phase, the algorithm identifies the set of k most centrally located points. The pseudo-code of the algorithm is shown in Algorithm 1. The inputs to the algorithm are the data matrix *data*, the number of central points to locate k and the distance function *dist*. `BUILD` first selects the most centrally located point (lines 2-6), and then tries to find the other $k-1$ points that would minimize the sum of distances to nearest central point (lines 7-19). It then returns the selected points.

Algorithm 1 `BUILD` phase of *Partitioning Around Medoids*

```
1:  $selected \leftarrow \emptyset$
2:  for all $i \in data$ do
3:      $cost_i \leftarrow \sum_{j \in data} d(i,j)$
4:  end for
5:  $best \leftarrow \arg\min_i cost_i$
6:  $selected \leftarrow \{best\}$
7:  for $c = 2$ to $k$ do
8:      for all $i \in data, i \notin selected$ do
9:          $savings_i \leftarrow 0$
10:         for all $j \in data, i \neq j, i \notin selected$ do
11:             $D_j \leftarrow min_{s \in selected} d(s,j)$
12:             if $D_j > d(i,j)$ then
13:                 $savings_i \leftarrow savings_i + D_j - d(i,j)$
14:             end if
15:         end for
16:     end for
17:     $best \leftarrow \arg\max_i savings_i$
18:     $selected \leftarrow selected \cup \{best\}$
19: end for
20: return $selected$
```

During the `SWAP` phase, the algorithm considers swapping a selected point i with a non-selected point j to further reduce the distance to nearest central point. The pseudo-code is shown in Algorithm 2. It takes in as input the output of `BUILD` (*selected*), *data*, and the distance function d. This algorithm computes the cost of swapping each selected data point s with some other point i. To do this, it looks at every other point j and sets a cost value based on the following distances:

- $d(j,s)$, where s is the representative point we're thinking of swapping, and j is an arbitrary non-representative point.
- $d(i,j)$, which is the distance between i, the point we're thinking of using as representative instead of s, and j
- D_j, which is the distance between j and the closest representative point.

- E_j, which is the distance between j and the second closest representative point.

If the value of $cost_i$ is negative (line 22), then there is a net benefit in replacing the point s with j, so we perform a swap (line 23).

Algorithm 2 `SWAP` phase of *Partitioning Around Medoids*

```
 1: for all $s \in selected$ do
 2:     for all $i \in data, h \notin selected$ do
 3:         $cost_i \leftarrow 0$
 4:         for all $j \in data, j \notin selected, j \neq h$ do
 5:             $currentcost \leftarrow 0$
 6:             $D_j \leftarrow min_{t \in selected} d(t, j)$
 7:             $BestRepresentative_j \leftarrow \arg\min_{t \in selected} d(t, j)$
 8:             $E_j \leftarrow min_{t \in \{selected \setminus BestRepresentative_j\}} d(t, j)$
 9:             if $d(s, j) > D_j$ and $d(i, j) > D_j$ then
10:                 $currentcost \leftarrow 0$
11:             else if $D_j = d(j, s)$ and $d(i, j) < E_j$ then
12:                 $currentcost \leftarrow d(i, j) - d(j, s)$
13:             else if $D_j = d(j, s)$ and $d(i, j) \geq E_j$ then
14:                 $currentcost \leftarrow E_j - D_j$
15:             else
16:                 $currentcost \leftarrow d(i, j) - D_j$
17:             end if
18:             $cost_i \leftarrow cost_i + currentcost$
19:         end for
20:     end for
21:     $best \leftarrow \arg\min_i cost_i$
22:     if $cost_{best} < 0$ then
23:         $selected \leftarrow selected \cup \{best\} \setminus \{s\}$
24:     end if
25: end for
26: return $selected$
```

3.4.2 Time and Space Complexity of Partitioning Around Medoids

In this section, we will analyze the time and space complexity of the *Partitioning Around Medoids* algorithm. This discussion motivates the creation of the approximation algorithm which is used in the thesis. We will assume that the data has n data points and f features, and we are tasked with finding k clusters.

It is traditional for clustering algorithms to use a *distance matrix* D where $D(i, j)$ gives the distance between data points i and j. This leads to huge savings in time because we frequently need the distance between data points. Building the distance matrix has the following costs:

- Normalizing the data takes $O(nf)$ time because we need to first look at every value of every feature to compute the statistics like mean absolute distance and then update each value to its standardized score. If update is performed in-place, this step requires $O(1)$ space.

- Computing the distance between any two points takes $\Theta(f)$ time because we consider each feature in the two data points. This is done for each pair of points in the data set, so we require $\Theta(n^2 f)$ time. This requires an additional $\Theta(n^2)$ storage.

So, computing the distance matrix takes $O(nf + n^2 f) = O(n^2 f)$ time, but we can now answer every distance query in $O(1)$ time.

We now consider the cost of the `BUILD` phase. Selecting the initial point (lines 2-4) requires computing the sum of distance from a point to every other point. This takes $O(n^2)$ time. We then consider every pair of non-selected points and compare their distance to the distance to the nearest selected point. The actual act of computing distances and comparing costs $O(1)$ time per pair because we use the precomputed distance matrix. However, this computation is done for each of the $O(n^2)$ pairs and repeated for each value of k. Thus, the total time cost of `BUILD` is $O(n^2 k)$.

Similarly, the `SWAP` phase performs a distance comparison computation, which costs $O(1)$, for each of the $O(n^2)$ pairs. This computation is repeated for each of the $O(k)$ selected poitns. Thus, the total cost is $O(n^2 k)$.

Thus, the total time complexity of the *Partitioning Around Medoids* algorithm is $O(n^2 f) + O(n^2 k) + O(n^2 k) = O(n^2 f + n^2 k)$. We require $O(n^2)$ space to store the distance matrix[6]. This is an unbearably high cost on a dataset of our size.

[6]If we choose not to compute the distance matrix, each distance computation takes $O(f)$ time. So both `BUILD` and `SWAP` take $O(n^2 kf)$ time. Since $f \approx 400$ in our dataset, this is a significant burden

3.4.3 The Clustering Large Applications Algorithm

We now discuss *Clustering Large Applications* [28, 27] an approximation to *Partitioning Around Medoids* that has much better performance characteristics. The basic idea of the algorithm is to perform *Partitioning Around Medoids* on smaller randomly selected samples of the data.

The algorithm is described in Algorithm 3. It takes as input the data matrix *data*, the number of clusters to produce k, the distance function d, the number of samples to use *nsamples*, the size of each sample *sampsize*. It generates a random sample of *data* of size *sampsize* using some built-in procedure `RandomSample`, performs clustering on the sample, assigns the remainder of the points to the discovered medoids, and computes the quality of this clustering using some metric `ClusterQuality` (such as Silhouette Index discussed in 3.3.1). It then compares the quality of clustering of each of the samples and picks the best clustering assignment.

Algorithm 3 The *Clustering Large Applications* algorithm

```
1: for $s = 1$ to $nsamples$ do
2:     $sample \leftarrow$ RandomSample($data, sampsize$)
3:     $clusters \leftarrow$ PartitionAroundMedoids($sample, k, d$)
4:     for all $x \in data, x \notin sample$ do
5:         assign $x$ to the closest medoid from $clusters$
6:     end for
7:     $quality_s \leftarrow ClusterQuality(clusters)$
8: end for
9: $best_s \leftarrow \arg\max_s quality_s$
10: return $best_s$
```

We now analyze the performance of *Clustering Large Applications.* There are four steps in the inner loop (lines 2-7), whose costs are:

Sample Generation which requires $\Theta(sampsize)$ time and $\Theta(sampsize)$ space.

Partition Around Medoids which requires $\Theta(sampsize^2 f + sampsize^2 k)$ time and $\Theta(sampsize^2)$ space.

Assigning Points to Closest Medoid requires distance computation ($\Theta(f)$ time) between each medoid $O(k)$ and each point $O(n)$. Total cost is $O(nkf)$.

Computing Cluster Quality whose cost depends on the exact quality metric used. With the recommended Silhouette metric, it takes $O(n)$ time.

So, the total cost of the inner loop is $O(sampsize + sampsize^2 + nkf + n) = O(sampsize^2 + nkf)$ time. This loop is performed *nsamples* time, so total cost is $O(nsamples \cdot (sampsize^2 + nkf))$ time. The time is now quadratic in sample size instead of quadratic in size of entire data set!

Of course, this improvement didn't come for free. The algorithm is no longer deterministic since it relies on randomly generated samples. It is also an approximation to *Partitioning Around Medoids* since we pretend that the clustering found from the sample is the clustering of the entire dataset. In this thesis, we do the following to minimize the approximation error:

- Use large samples. We used the 30,000, which was the largest sample size that we could work with.
- Use high value of nsamples. We used 50 in order to be reasonably confident that the clusters that were discovered weren't needlessly bad.

3.5 Summary

This chapter used a simple example to introduce *clustering*. We identified the critical components of clustering: the *distance metric*, the number of clusters to identify, and the algorithm to perform the clustering and discussed each in detail. We started by formalizing the notion of metric and introduced some simple metrics. These metrics were successively improved to handle missing values, heterogeneous features and different scales. We briefly discussed the idea of a clustering validation metric, and defined the *Silhouette Index* which will be used for model selection. We then discussed the limitations of the venerable *k-means algorithm* that motivated the development of the *partitioning around medoids algorithm*. We analyzed the performance of partitioning around medoids saw that it was too high for a dataset of our size. Finally, we developed Clustering Large Applications, an approximation to *Partitioning Around Medoids* based on randomized sampling.

Chapter 4

Results

This chapter discusses the creation and validation of the clustering model. We begin by discussing the different parameters of the clustering algorithm and the selection of the number of clusters based on two measures of cluster quality: the *Silhouette Index* [28] and the *Gap Statistic* [40]. However, we aren't simply interested in grouping cases that are similar in some abstract feature space; we want to connect these groups into external features such as outcome measures. Therefore, we investigate the extent to which the cases in the clusters found by our algorithm are enriched for specific outcomes such as Survival, Glasgow Coma Scale and Critical Heart Rate Events.

4.1 Model Selection : Parameters

The Clustering Large Applications algorithm described in Section 3.4.3 has the following parameters:

- k, the number of clusters to produce
- d, the distance function
- $nsamples$, the number of random samples of the data to use
- $samplesize$, the size of each of those random samples

The parameters *samplesize* and *nsamples* only affect the convergence of results and were set to 30000 and 50 respectively, which were the largest values that were computationally tractable. As discussed in Section 3.4.3, the Clustering Large Applications has a time and space complexity that grows quadratically with *samplesize*, and we found that values of *samplesize* larger than 30000 exhausted the 28 Gigabytes of memory available on our computer. We set *nsamples* to 50 because we found that 50 repetitions of clustering on a *samplesize* of 30000 was sufficient to ensure a consistency in the medoids discovered.

In Chapter 3, we described a distance function that was capable of dealing with missing data, data of different types, and numeric data of different scales. This distance function is itself parametrized by distance functions for numeric data types, symmetric nominal variables and asymmetric nominal variables. In this thesis, we picked three simple and ubiquitous distance metrics: Euclidean distance for numeric, Simple Matching coefficient for symmetric nominal and Jaccard's distance for asymmetric nominal variables.

In order to determine the appropriate value for the final parameter, k, we first considered the hypothesis space. Theoretically, k can take on any value between 1 and N, the number of data points in the dataset. Practically, we are interested in smaller values of k for the following reasons:

Compression The smaller the value of k, the more *compressed* the dataset is. One of the primary goals of the thesis is to ease reasoning by reducing the size of a dataset that has to be considered without losing salient features. Thus, large values of k are explicitly against the stated goals of this thesis.

Robustness Since the dataset is noisy, larger values of k might result in clustering models that are overfit to the noise present in the dataset.

Learnability We want the resulting labels from clustering to be usable by subsequent models. An output of a large number of labels increases the state space and hampers learnability of these models.

In this thesis, we considered a values of k less than 100. For comparison, similar work on patient state discovery on a smaller dataset discovered 10 clusters in their dataset [11].

In order to determine the best value of k, we used the *Silhouette Index* criteria (3.3.1) first and then validated the choice using the *Gap Statistic*. Both Silhouette and Gap measure only the structural soundness of the discovered clusters and do not consider the fitness of clusters towards any specific outcome. This is in line with our aim to discover "natural" grouping of data that is generally useful across a spectrum of outcomes.

4.2 Model Selection using the Silhouette Index

The Silhouette Index is introduced in Section 3.3.1. Recall that the Silhouette Index is a quantity that determines goodness of clustering by comparing the average dissimilarity between a point and other points in the same cluster to the average dissimilarity between the point and the next best cluster. Unlike most other metrics for cluster quality, values of the Silhouette Index do not increase monotonically with k; it penalizes large values of k as well as small values. Values of the Silhouette Index range between -1 and 1 with larger values indicating better clustering. The creators of the Silhouette Index suggest the following interpretation of the value [28]:

0.71–1 Strong structure discovered by clustering

0.51–0.70 Reasonable structure discovered

0.26–0.50 Weak, possibly artificial, structure discovered

-1–0.25 No substantial structure discovered

Figure 4-1 shows the value of the Silhouette Index for clusters discovered for values k between 2 and 100. The clustering was performed on the dataset described in Chapter 2. The other parameters of the clustering algorithm were set as described in the previous section. Table 4.1 shows the best values of the Silhouette Index and

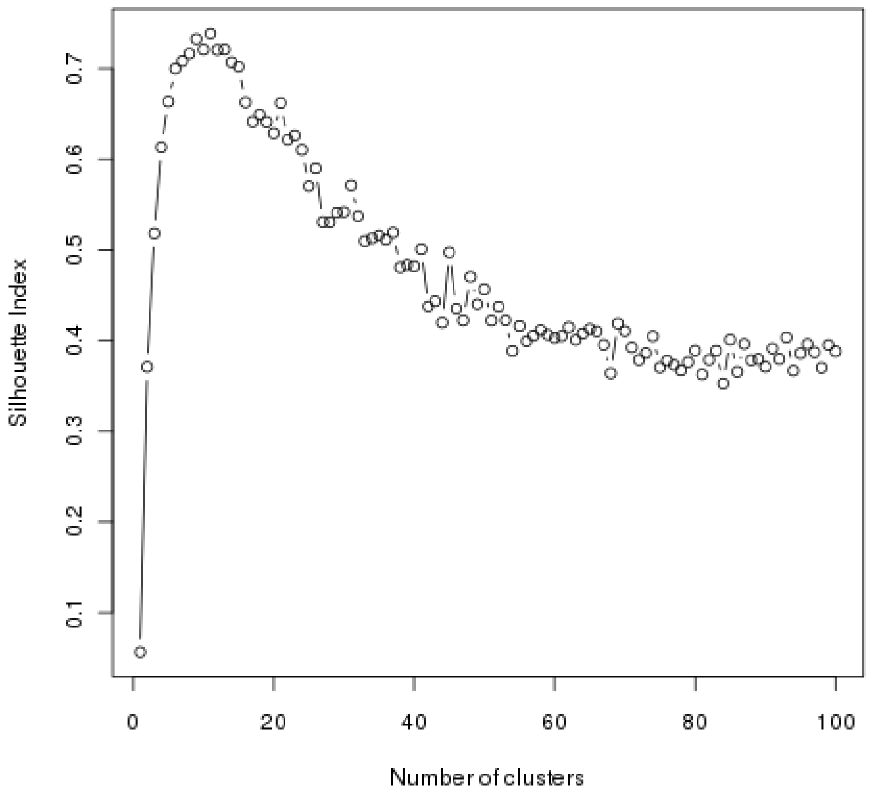

Figure 4-1: The Silhouette Index values for clusters discovered with different settings of k, the number of clusters. Higher values indicate better clustering. The best value is at $k = 11$.

Figure 4-2 shows the values of the index for k less than 20. The best value is 0.7384 at $k = 11$, which suggests the discovery of a strong structure. However, the values between 7 to 15 all have scores above 0.70, suggesting that the dataset contains between 7 and 15 clusters.

Figure 4-3 shows the size of each cluster. We see that the clustering has produced three large clusters and five smaller ones.

4.3 Model Validation using the Gap Statistic

In order to verify that the value of k obtained by optimizing the value of the Silhouette Index, we used the *Gap Statistic* [40], a different method to determine the correct

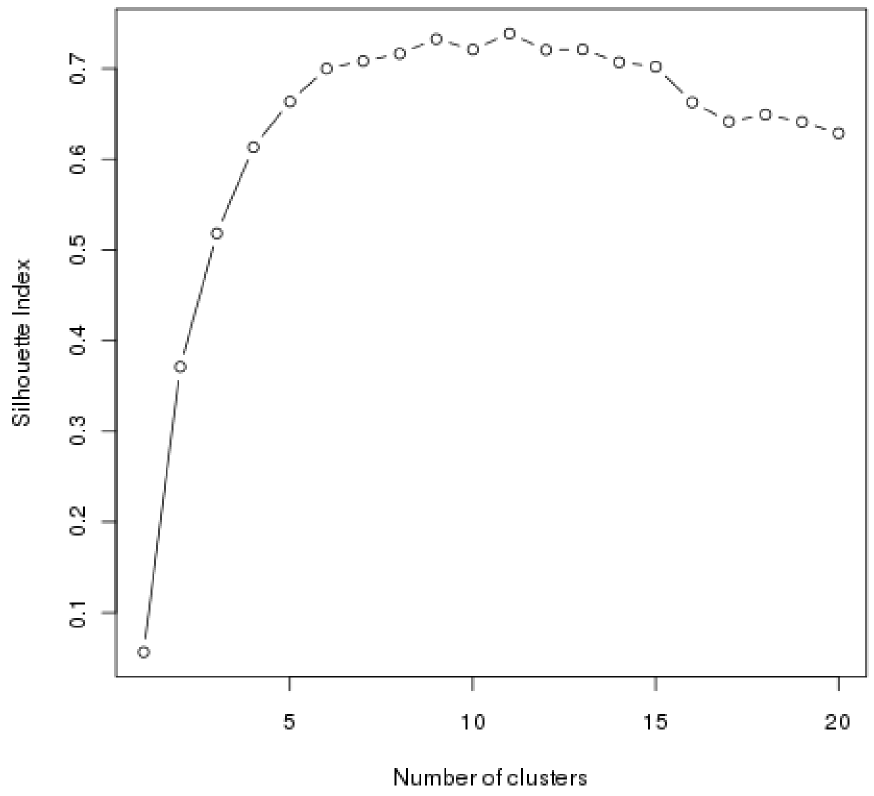

Figure 4-2: Silhouette Index values for clusters focusing on k. This plot is otherwise the same as Figure 4-1 except that it shows values of k between 1 and 20.

number of clusters in a dataset.

The Gap Statistic is based on the observation that the within-cluster dissimilarities (W_k) initially decreases monotonically as k increases, but starts flattening after a certain value of $k = k_0$. While the exact value of k_0 depends on the dataset, this value represents the point beyond which quality of clustering cannot be improved dramatically simply by adding more clusters. It is commonly accepted that this value of k represents the number of clusters in the dataset [22, 40]. The Gap Statistic is a method of determining the k where the W_k curve starts to flatten.

More formally, the pooled within-cluster dissimilarity is defined as follows:

$$W_k = \sum_{r=1}^{k} \frac{1}{2n_r} D_r \tag{4.1}$$

k	Silhouette Index
7	0.7082571
8	0.7164445
9	0.7326193
10	0.7210056
11	0.7384862
12	0.7205812
13	0.7211385
14	0.7069354
15	0.7021224
16	0.6628145

Table 4.1: The values of k that have the largest Silhouette Index. Largest possible value is 1, and values larger than 0.70 suggest the discovery of strong structure.

where n_r is the number of items in the r^{th} cluster and D_r is the sum of distances between all pairs of points in the r^{th} clusters.

The Gap Statistic first normalizes values of $log(W_k)$ by comparing it with $log(W_k^*)$, its expected value under a null reference distribution. For computational simplicity, the authors suggest using the uniform distribution over a box that is aligned with the principal components of the data instead of the Maximum Likelihood Estimate of the data. In order to obtain the estimate, B bootstrap samples of data from the reference distribution are generated and clustered to obtain the $log(W_{kb}^*)$, the estimate of the $log(W_k^*)$ for the b^{th} bootstrap sample. Then, the Gap Statistic is computed using the following formula:

$$Gap(k) = \frac{1}{B}(\sum_b log(W_{kb}^*)) - log(W_k) \tag{4.2}$$

The authors suggest selecting the number of clusters in the dataset, $\hat{k}$ using *Tibshirani's criteria*, which is defined as:

$$\hat{k} = \arg\min_k Gap(k) \geq Gap(k+1) - \sqrt{1+\frac{1}{B}}sd(log(W_{kb}^*)) \tag{4.3}$$

In other words, select the smallest k where the Gap is larger than the Gap of the $k+1$. The last term in the equation are error terms that account for error accrued

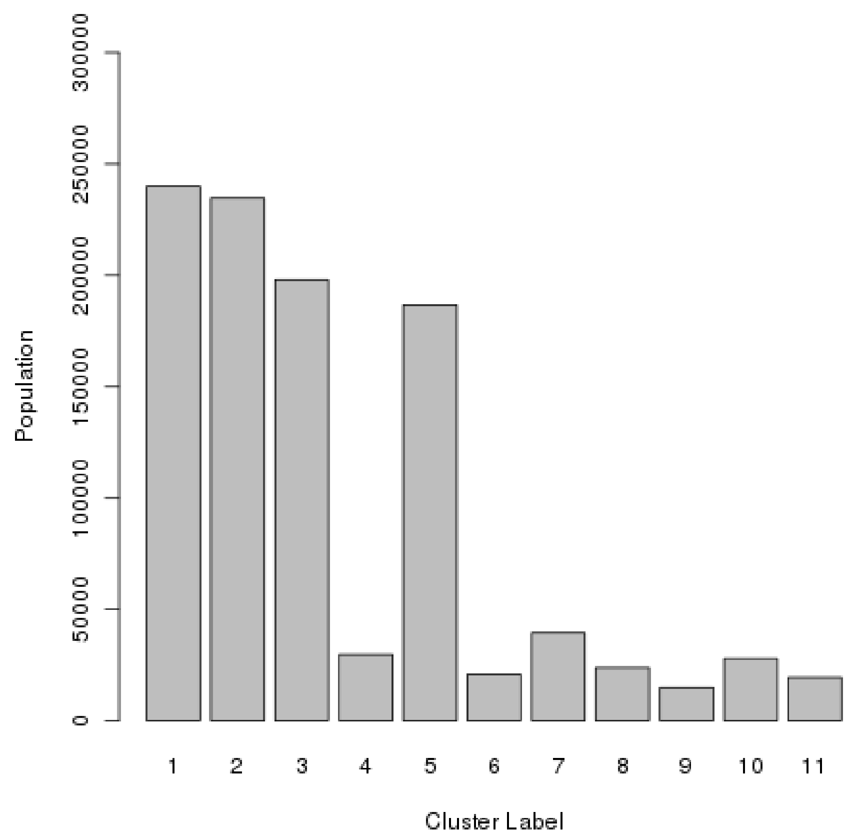

Figure 4-3: Number of items in each cluster.

during bootstrap estimation.

We computed the values of the Gap statistic on our dataset for k between 1 and 100 with number of bootstrap samples $B = 50$. The results are shown in Figure 4-4 and Figure 4-5. Using Tibshirani's criteria, the best value of k is 11, the same value picked using the Silhouette Index.

4.4 Model Validation by Verifying Enrichment

In this section, we analyze the clusters discovered for biological significance by verifying enrichment towards specific outcomes. In other words, we compare the probability of an outcome, such as survival, in the entire dataset[1] with the probability of the same

[1]This corresponds to the null hypothesis that there is only 1 cluster in the data

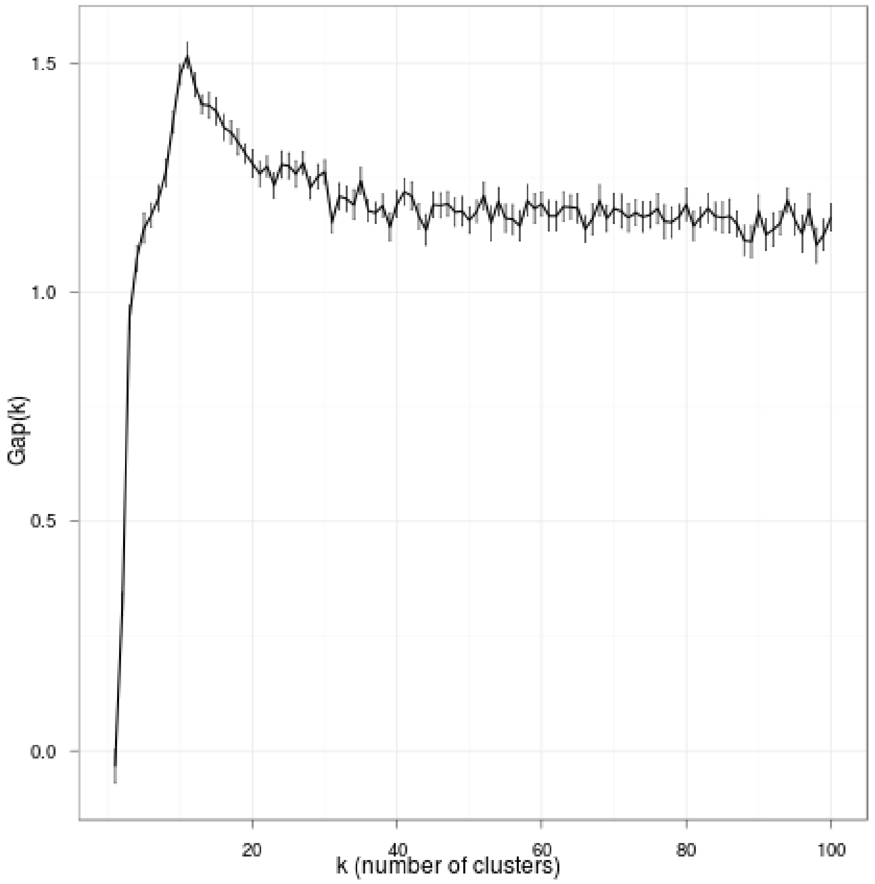

Figure 4-4: Values of the Gap statistic for k between 1 and 100. Tibshirani's Criteria recommends picking the k where $Gap(k)$ is greater than $Gap(k+1)$ (after accounting for estimate errors). In our dataset, this corresponds to $k = 11$

outcome in the different clusters.

We say that a cluster is enriched for an outcome when it has a higher probability than the entire dataset[2]. We say that a cluster is depleted towards an outcome when it has a lower probability. Enrichment and depletion of clusters suggest that the clustering results are biologically significant because a uniform random partition of the data into clusters would not affect the probability of an outcome.

More concretely, to verify enrichment or depletion we first compute the baseline value of the dataset using the following formula:

[2]For statistical significance, the number of points in the cluster has to be sufficiently large

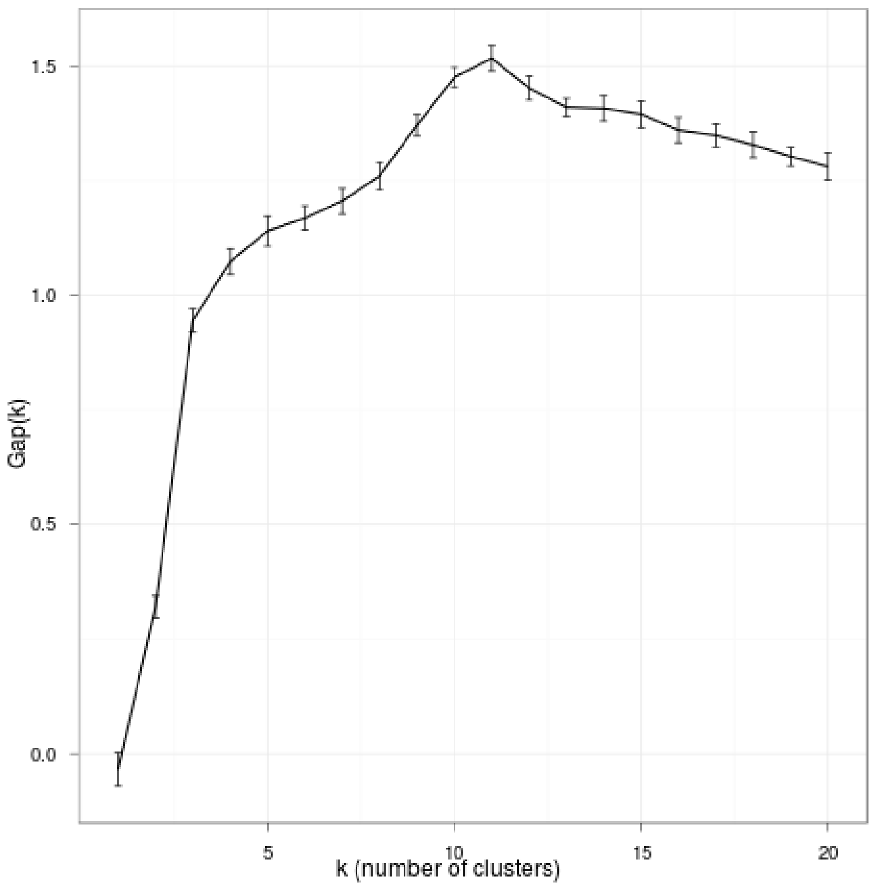

Figure 4-5: Values of the Gap statistic for k between 1 and 20. This figure is the same as Figure 4-4 except that it shows values between 1 and 20.

$$probability(outcome) = \frac{\text{Number of data points belonging to patients with the outcome}}{\text{Total data points in the dataset}} \tag{4.4}$$

We then compute the probabilities per cluster:

$$probability(outcome|cluster = k) = \frac{\text{Number of data points in cluster k with outcome}}{\text{Total data points in cluster k}} \tag{4.5}$$

The two probability values are then compared to see if the cluster probabilities are markedly higher (enrichment) or lower (depletion). We will now discuss the result of testing for enrichment for three outcomes. We have arbitrarily labeled the 11 clusters are with numbers $1, \cdots, 11$.

4.4.1 Survival

As in [24], we say that patients have "survived" if they are alive 30 days after being discharged from the ICU. The value of the baseline for survival rate in our dataset was found to be 0.8329. This baseline along with values for each of the 11 clusters are shown in Figure 4-6.

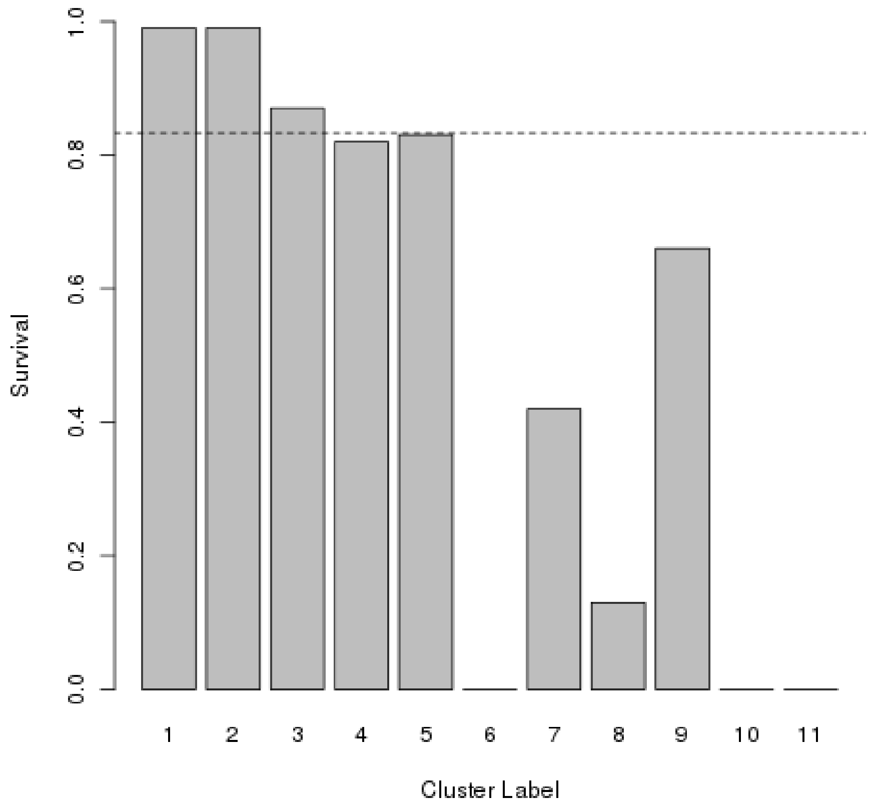

Figure 4-6: Fraction of points in each cluster belonging to patients who were alive after 30 days of being discharged from the ICU. The dashed line shows the baseline for the entire dataset.

The survival data has the following characteristics:

- Clusters 1 and 2 are composed exclusively of data points belonging to patients that survived. This is significant because clusters 1 and 2 are quite large.

- Clusters 6, 10 and 11 are composed exclusively of data points belonging to patients that expired
- Clusters 7, 8, and 9 show survival rates that are significantly lower than baseline
- Clusters 3, 4, and 5 have survival rates that are close to baseline

4.4.2 Glasgow Coma Scale

The Glasgow Coma Scale [39] measures the patient's consciousness by testing eye, verbal and motor responses. The score varies between 3 and 15 with lower scores indicating greater severity. The scores are typically interpreted as follows:

- Severe Brain Injury: GCS ≤ 8
- Moderate Brain Injury: GCS $9 - 12$
- Minor Brain Injury: GCS ≥ 13

The expected value of GCS in our dataset was 11.34[3]. The expected value of GCS for each of the 11 clusters are shown in Figure 4-7.

The following clusters contain values that are significantly different from baseline:

- Clusters 1 and 2 both have expected GCS value of over 14, suggesting a population with minimal brain injuries
- Clusters 6, 8, 10 and 11 all have expected GCS of 3, which is the most severe value.
- Clusters 7 and 9 have GCS values close to 7 suggesting severe brain injury

[3]The value was missing in 51,831 data points. These points were omitted during the computation of expectation.

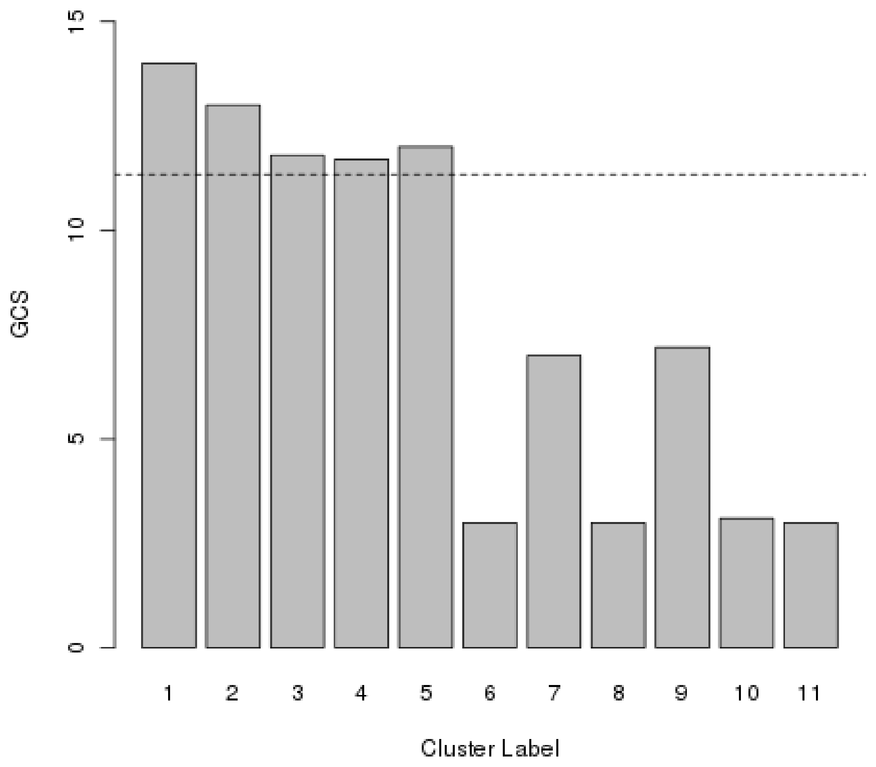

Figure 4-7: Expected value of Glasgow Coma Scale (GCS) in each cluster. The dashed line shows the baseline for the entire dataset.

4.4.3 Number of Critical Heart Rate Events in Past 24 Hours

Finally, we analyze the distribution of the number of critical heart rate events in the past 24 hours (abbreviated HRCritEvents). A heart rate value is said to be critical if it is:

- Less than 50 beats per minute, or
- Systolic Arterial Pressure is less than 60 mmHg, or
- Mean Arterial Pressure is less than 50 mmHg

The distribution of critical events is very long tailed with the mean at 6.665, third quartile at 5 and the maximum at 223. The expected values of HRCritEvents for each of the 11 clusters are shown in Figure 4-8.

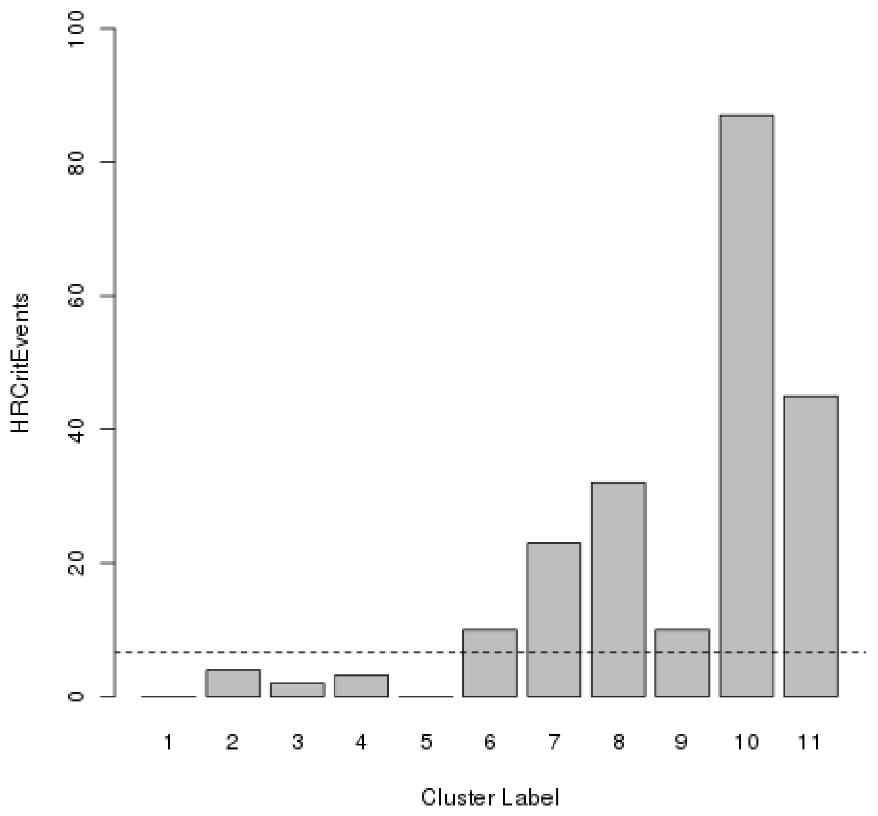

Figure 4-8: Expected Number of Critical Heart Rate Events in the past 24 hours. The dashed line shows the baseline for the entire dataset.

The following clusters contain values significantly different than baseline:

- Clusters 1 and 5 have an expected value of 0 and contain points that had very few or no critical events

- Clusters 2, 3, and 4 have expected values less than 5, and contain points with few critical events.

- Clusters 7, 8, 10 and 11 have expected values well above the baseline. In particular, the expected value of Cluster 10 is over 80 critical heart rate events in the past 24 hours.

4.5 Conclusion

We set the number of clusters to be 11 by optimizing the Silhouette Index as suggested in [28]. This clustering assignment has a Silhouette Index of 0.7384 which suggests strong structure discovery. This choice of the number of clusters was verified by using the Tibshirani's Criteria of Gap Statistic. Finally, we compared the expected value of Survival rate, Glasgow Coma Scale, and Number of Critical Heart Rate Events in the past 24 hours in each cluster to the value in the entire dataset. We found clusters with values significantly different from baseline, suggesting that our clustering had done better than a simple random partition of the data into 11 clusters.

Chapter 5

Conclusion

This thesis discussed the use of a non-parametric clustering technique to identify abstract patient states. As a conclusion, we will first review the work done in the thesis and then discuss areas of further research.

5.1 Summary of Contributions

We started the thesis by discussing the problem of reasoning with large, high-dimensional, noisy and heterogeneous datasets. We stated our goal of producing a compressed version of the dataset that still retained the salient features. We described the success of clustering, an unsupervised learning technique, in achieving the goal in a variety of fields.

In Chapter 2, we delved deeper into the MIMIC II database. We described the different types of data available and the features that were eventually selected for the dataset. We then discussed the problems with the dataset such as large size, high dimensionality, sparsity, and noise along with some measures to mitigate the problems such as feature selection and holding.

In Chapter 3, we identified the components of partitional clustering: metric, parameter for number of clusters, and the clustering algorithm. We developed a metric capable of dealing with the missing data, heterogeneous features and measurements in different scales. We discussed the use of clustering indices to measure quality of clustering and the use of such indices to select the right number of clusters. We then discussed a robust and computationally efficient algorithm to perform clustering.

Chapter 4 discussed the results of clustering with the best k as determined through Silhouettes. The choice of best k was verified using Gap Statistic, a different metric. We then evaluated the discovered clusters for biological significance by checking for enrichment towards survival, Glasgow Coma Scale values, and Number of Critical Heart Rate Events. In each case, we found clusters whose values were significantly higher and lower than the baseline.

To summarize, this thesis developed a fast, scalable clustering capable of dealing with large, high-dimensional, noisy, incomplete data with mixed variables. The results obtained by this method showed biological significance.

5.2 Future Work

We will now discuss avenues for future research in the areas of data processing, dissimilarity calculation and clustering.

5.2.1 Data Processing

As mentioned in Chapter 2, we only used the nurse-verified clinical dataset portion of the MIMIC II dataset in this thesis. Incorporating the nursing notes data using techniques like topic modeling [5] and the waveform data using waveform clustering [35] would allow us to discover much richer structures.

Although the MIMIC dataset is a time series, we didn't exploit this property in any way other than to perform holding to fill in missing values. We could use time series alignment to compute clusters at different times in a patient's stay to get a temporal model of the patient's state evolution.

5.2.2 Dissimilarity Computation

The Euclidean distance used in the thesis is one of many possible metrics for numeric dissimilarity. Unlike the Mahalanobis distance [29], it does not incorporate the information on the correlations between the different features of the dataset. As a result, the distances obtained by Euclidean distance are not corrected for correlations between different variables. Equivalently, this means that the discovered clusters correspond to hyperspheres around the central object instead of the more general hyperellipsoids.

Furthermore, we have assigned equal weights to features of the dataset even though they differ in their predictive power. For example, the amount of Fentanyl administered to a patient can predict patient survival time better than the amount of Dobutamine administered [23]. We could perform the same type of analysis on other hazardous outcomes such as organ failures to obtain a better idea about the predictive power of different features. This information could be used to weight the different features in the distance computation.

5.2.3 Clustering

Experience with clustering in different fields [28, 15, 22] has shown that the particular technique used to discover clusters can have a large effect on the quality of clusters discovered. This is in contrast to supervised learning, where experience has shown that the particulars of the algorithm do not dramatically change the quality of model learned. Thus, even though the algorithm used in the thesis has demonstrably produced good clusters, it is hard to claim that it has discovered the best clusters. Experimentation with different clustering algorithms, particularly those that make radically different assumptions, would be useful. In the following paragraphs, we will discuss a few such algorithms.

The algorithm in the thesis assigned each data point to a single cluster, so it belongs to a class of *hard clustering* algorithms. In contrast, *fuzzy-clustering* algorithms assign a fractional *membership value* of each point to a cluster, which specifies how well a point can be said to belong to a specified cluster. Since these algorithms are free to assign points to multiple clusters, they can perform better than hard clustering algorithms [28, 15]. However, these algorithms are generally much less efficient than hard clustering algorithms, which is a problem in a large dataset such as ours.

Similarly, the algorithm in the thesis is *partitional* since it partitions the dataset into a fixed number of clusters. *Hierarchical* clustering algorithms, on the other hand, produce a hierarchy of clusters with all the data points in one end and a single cluster containing them all in the other. Since finding the best possible hierarchical grouping in an arbitrary dataset is NP-complete [3, 12] greedy algorithms are used to generate the hierarchy. As a result, the k clusters obtained from a hierarchical clustering algorithm could be worse than the k clusters discovered by a partitional clustering algorithm [28, 12]. Furthermore, hierarchical algorithms are worse with regard to both time and space complexity than partitional algorithms, and getting them to work with a large dataset like ours will require some ingenuity.

Co-clustering [21, 14] algorithms, which perform clustering on features (columns) and data (rows) simultaneously, have been successfully used to discover clusters in gene expression data [9]. A co-clustering algorithm would, in theory, automatically perform feature selection in the "best" way as far as clustering the data is concerned, making it easier to work with a high-dimensional dataset like ours. Another way to reduce the dimensionality is to use *Spectral clustering* algorithms [33] which perform clustering with respect to the eigenvectors corresponding to the top few eigenvalues of the distance matrix. These algorithms have yielded good results in computer vision [30] and could do the same here.

Finally, ideas from sketching and streaming algorithms [16] could be used to derive approximate clustering algorithms for use with large datasets. For example, [6] performed clustering on 30 million documents using sketches of documents. Similarly, [20] describe an approximation algorithm for k-means and k-median clustering based on *coresets* [1].

Bibliography

[1] Pankaj K. Agarwal, Sariel Har-peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. *combinatorial and computational geometry*, 2004.

[2] A. Azzalini and A.W. Bowman. A look at some data on the old faithful geyser. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 39(3), 1990.

[3] Therese Biedl, Brona Brejova, Erik D. Demaine, Angele M. Hamel, and Tomas Vinar. Optimal arrangement of leaves in the tree representing hierarchical clustering of gene expression data. Technical report, University of Waterloo, 2001.

[4] David Blackwell and James B. MacQueen. Ferguson distributions via polya urn scheme. *Annals of Statistics*, 1(2), March 1973.

[5] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. In *Neural Information Processing Systems*, pages 601–608, 2001.

[6] Andrei Z. Broder, Steven C. Glassman, Mark S. Manasse, and Geoffrey Zweig. Syntactic clustering of the web. *Computer Networks and Isdn Systems*, 29:1157–1166, 1997.

[7] George Casella and Roger L. Berger. *Statistical Inference*. Duxbury Press, second edition, 2001.

[8] Leo Anthony G Celi. Localized customized mortality prediction modeling for patients with acute kidney injury admitted to the intensive care unit. Master's thesis, Massachusetts Institute of Technology, 2009.

[9] Yizong Cheng and George M. Church. Biclustering of gene expression. *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, 2000.

[10] Gari D. Clifford, Daniel J. Scott, and Mauricio Villarroel. User guide and documentation for the mimic ii database. *http://mimic.mit.edu/documentation.html*, 2011.

[11] Mitchell J. Cohen, Adam D. Grossman, Diane Morabito, M. Margaret Knudson, Atul J. Butte, and Geoffrey T. Manley. Identification of complex metabolic states in critically injured patients using bioinformatic cluster analysis. *Critical Care*, 14, 2010.

[12] Sanjoy Dasgupta and Philip M. Long. Performance guarantees for hieararchical clustering. *Elsevier Science*, 2010.

[13] Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1), 1977.

[14] Meghana Deodhar and Joydeep Ghosh. Scoal: A framework for simultaneous co-clustering and learning from complex data. *ACM Transactions on Knowledge Discovery from Data*, 2009.

[15] Brian S. Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. Oxford University Press, fourth edition, 2001.

[16] Philippe Flajolet and G. Nigel Martin. Probabilistic counting for data base applications. *Journal of Computer and System Sciences*, 1985.

[17] Raffaele Giancarlo, Davide Scaturro, and Filippo Utro. Computational cluster validation for microarray data analysis: experimental assessment of clest, consensus clustering, figure of merit, gap statistics, and model explorer. *BMC Bioinformatics*, 9, 2008.

[18] J.C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 1971.

[19] Frank R. Hampel, Elvezio M. Ronchetti, Peter J. Rousseeuw, and Werner A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley-Interscience, first edition, 2005.

[20] Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *ACM Symposium on Theory of Computing*, pages 291–300, 2004.

[21] John A Hartigan. Direct clustering of a data matrix. *journal of the american statistical association*, 1972.

[22] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, second edition, 2009.

[23] Caleb W. Hug. Predicting the risk and trajectory of intensive care patients using survival models. Master's thesis, Massachusetts Institute of Technology, 2006.

[24] Caleb W. Hug. *Detecting Hazardous Intensive Care Patient Episodes Using Real-time Mortality Models*. PhD thesis, Massachusetts Institute of Technology, 2009.

[25] Paul Jaccard. Nouvelles recherches sur la distribution florale. *Bulletin de la Socit Vaudoise des Sciences Naturelles*, 1901.

[26] Leonard Kaufman and Peter J. Rousseeuw. Clustering by means of medoids. In Y. Dodge, editor, *Statistical Data Analysis based on the L1 norm*, chapter 2. Elsevier, 1987.

[27] Leonard Kaufman and Peter J. Rousseeuw. Pattern recognition in practice ii. In Edzard S. Gelsema and Laveen N. Khanal, editors, *Clustering large data sets*, chapter 3. Elsevier, 1987.

[28] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis.* Wiley, 1989.

[29] P. C. Mahalanobis. On the general distance in statistics. *National Institute of Sciences of India*, 1936.

[30] Jitendra Malik, Serge Belongie, Thomas Leung, and Jianbo Shi. Contour and texture analysis for image segmentation. *Perceptual Organization for Artificial Vision Systems*, 2000.

[31] Glenn W. Milligan and Martha C. Cooper. An examination of procedures for determining the number of clusters in a data set. *psychometrika*, 50(2):159–179, 1985.

[32] James R. Munkres. *Topology.* Prentice Hall, second edition, 2000.

[33] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2001.

[34] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Computational and Applied Mathematics*, 1987.

[35] C. A. Rowe, R. C. Aster, B. Borchers, and C. J. Young. An automatic, adaptive algorithm for refining phase picks in large seismic data sets. *Bulletin of the Seismological Society of America*, 2002.

[36] Walter Rudin. *Principles of Mathematical Analysis.* McGraw-Hill, third edition, 1976.

[37] M. Saeed, G. Lieu, C.and Raber, and R. G. Mark. MIMIC II: a massive temporal ICU patient database to support research in intelligent patient monitoring. *Computers in Cardiology*, 29:641–644, September 2002.

[38] Mark Steyvers and Tom Griffiths. Probabilistic topic models. In Thomas K. Landauer, Danielle S. McNamara, Simon Dennis, and Walter Kintsch, editors, *Handbook of Latent Semantic Analysis.* Psychology Press, 2007.

[39] Graham Teasdale and Bryan Jennett. Assessment of coma and impaired consciousness: A practical scale. *Lancet*, 1974.

[40] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B*, 63(2):411–423, 2001.

[41] Joseph Zubin. A technique for measuring like-mindedness. *Journal of Abnormal and Social Psychology*, 1938.