ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Massachusetts
Institute of
Technology

Master Thesis

Alexander Waldin

Learning Blood Pressure Behavior From Large Blood Pressure Waveform Repositories and Building Predictive Models

Supervisors:	Prof. Gerhard Tröster Dr. Una-May O'Reilly
Advisors:	Dr. Kalyan Veeramachaneni

June 10, 2013

Abstract

We consider the problem of predicting the blood pressure of a general patient in an intensive care unit using only information contained in his blood pressure signal. For our dataset we build a “Beat Database” from over 6,000 arterial-blood-pressure waveform records. To make predictions we build both static and dynamic models. To build the static models we design a “Multi-Algorithm, Multi-Parameter Suite” that allows us to explore hundreds of different combinations of classification algorithms and parameter settings. Our dynamic model is a latent state-space model, and to build it we design a sophisticated Bayes network. We train and evaluate our models on 1000 arterial-blood-pressure waveform records and compare their performance against a baseline. We discover that the models' performance is not significantly better than the baseline and conclude that, to predict blood pressure for a general patient, further research is required in the form of creating models for specific subgroups of patients and fusing them together, as well as in the form of additional features other than mean arterial blood pressure extracted from arterial blood-pressure waveform records.

Acknowledgements

I would like to thank Will Drevo for his help in implementing the Multi-Algorithm, Multi-Parameter Suite and Franck Dernoncourt for his collaboration in designing and implementing the beat feature database. Furthermore, I am very grateful to both Una-May O'Reilly and Kalyan Veeramachaneni for their wonderful support and enthusiasm for my research. Finally, I would like to thank Gerhard Troester for giving me the opportunity to research and write this thesis here at MIT.

Contents

1 Introduction

1.1 Motivation

Patients with serious injuries or illnesses who require constant attention from medical staff are placed in an Intensive Care Unit (ICU). Because patients in intensive care are often unconscious or otherwise incapable of alerting medical staff if their condition deteriorates, they are monitored by bedside sensors that continuously record signals such as electrocardiogram and arterial blood pressure waveforms. If a patient enters a critical state while being monitored, an alarm is sounded to alert the medical staff. Such alarms, however, only allow doctors to react after the fact instead of helping them prevent these dangerous conditions in the first place.

The scientific community has collected data recorded by medical monitoring equipment attached to patients and created databases of medical information. These databases allow them to mine historical medical records and discover patterns therein that can be used to make statements about the future development of patients. Being able to foretell a patient's future state would allow doctors to administer preventive treatment, thus saving lives and improving the use of human and financial resources.

One specific goal that researchers have is to design algorithms and models using data from such databases that predict how the blood pressure of a patient in an ICU will change. Blood pressure prediction, and in particular the prediction of hypotension, is of great interest to doctors and ICU staff as hypotension can be caused by a wide variety of conditions and disorders. Being able to identify the patients that are at risk of developing low blood pressure would be beneficial in several ways:

- It would reduce ICU operation costs and increase operation efficiency.
  - Medical staff could focus on monitoring patients that are particularly at risk.
  - Prevention is often less expensive than intervention. Once a patient has entered a critical state, such as shock, an expensive life-saving intervention is necessary.
- Doctors would have additional information available to them when making decisions on how to treat patients.
- Knowing a patient's state can give their relatives and loved ones peace of mind.

We believe that it is possible to tap predictive information within medical record databases, allowing us to predict how the blood pressure of a patient will change. Because blood pressure is influenced by many different factors, it is necessary to build predictive models using a large and diverse set of data so that the model can correctly identify the condition of a new patient when predicting that patient's blood pressure. For example, a patient in a coma is more likely to have a different average blood pressure than a patient not in a coma. A model that was not created using data from patients in a coma will have difficulty predicting the blood pressure of a new patient if that new patient is in a coma.

2 Contributions

2.1 Novel Blood Pressure Prediction Problem

We explore the potential of predicting an ICU patient's blood pressure using only their blood pressure as a sensor input. The challenge here is to be able to predict the future state of the arterial blood pressure (ABP) itself with no clinical information or any other additional information from doctors. Whereas previous research used the ABP signal to predict specific medical conditions or, as is in the case of predicting an acute hypotensive episode (AHE), predict a condition that is a derivative of the ABP signal, we predict the future state of the ABP signal itself using only the ABP signal as an input. More specifically, we predict future blood pressure and not a medical event.

The reason for predicting only the future blook pressure is that it is very difficult to agree on the definition of a medical event in such a way that it can be both measured accurately and applied to a wide spectrum of patients. For example, the Physionet 2009 challenge defined a hypotension to be in a specific pressure range, but when predicting an AHE one must take into consideration that, depending on the individual, systolic and diastolic blood pressure can vary by as much as 30 to 40 mmHg over 24 hours. This means sensitivity to fluctuation varies by individual; for some, a small drop in blood pressure is harmless, while for others with naturally low blood pressure, a drop by 20 mmHg has been shown to cause transient hypotension [7]. By predicting the blood pressure signal itself, we do not make any assumptions about individuals. Instead, we provide additional information for a doctor to use when making a decision.

We acknowledge that there are many data sources available in addition to the ABP waveform, both in the form of waveforms other than the ABP signal as well as clinical data, that can be used to predict a patient's future blood pressure. Although incorporating these additional data sources into our models may improve them, we made the decision to use only ABP waveform data.

By focusing on predicting the blood pressure waveform and by restricting our data sources to the ABP signal, we contribute to the blood pressure prediction problem in that we:

- take into account the unique characteristics of each individual patient by making a prediction as to how his blood pressure will develop and not interpreting the development of his blood pressure as a medical event.
- test the limits of the information contained in the ABP waveform data with respect to predicting the future of a patient's blood pressure.
- design a universal system that is not only applicable anywhere, but also economic, because measuring a patient's blood pressure is straightforward and most ICUs around the world have this capability.

We formulate our problem mathematically in Section 4.

2.1.1 Feasibility

Although predicting blood pressure using only the ABP signal as an input is a very challenging problem, we hypothesize that we can achieve this for the following reasons:

- As discussed in Section 3.2.1, several studies have been able to predict AHEs, a medical event dependent on blood pressure, using only the ABP signal.

- Most studies that have used ABP signals have used trend data sampled at once a minute, the so called numerics record in the MIMIC-II waveform dataset. We use waveform data that was acquired by sampling the ABP signal at 125hz, and we hypothesize that this will yield better information than the trend data used by others.

- We use data from the entire MIMIC-II waveform dataset. Many studies have used records from far fewer patients, for example, the Physionet 2009 challenge described in Section 3.2.1 provided a training-set of only 60 patients. By using a large data set, we expect to have data from a wide range of different kinds of patients in different conditions, allowing us to make better predictions for new patients because our models are more likely to contain data from a patient that is similar to the new patient.

2.2 Beat Database

To build and evaluate predictive models we use the complete set of ABP data from the MIMIC-II waveform dataset. See Section 3.1.2 for an overview of the literature describing the MIMIC-II database and the MIMIC-II waveform dataset. Building and evaluating models requires us to transform the blood pressure data by extracting features from the signal waveform. As the size of the entire blood pressure data in the MIMIC-II database is over 3 terabytes, it is not feasible to perform feature extraction repeatedly when building models. For this reason we take the MIMIC-II ABP waveform dataset and transform it into a beat database. This beat database is constructed by running a beat onset detection algorithm on every waveform record and extracting contiguous segments of the waveform belonging to one beat. We then analyze these beat segments and either flag them as noisy or extract signal features from them. From the MIMIC-II waveform data we thus create a new dataset of beat features which we call the beat database. We describe the details of this process in Section 6. The beat database allows researchers to quickly build and evaluate models using data from the entire MIMIC-II ABP waveform dataset. It is our goal to make the beat database accessible to the scientific community in the future.

2.3 End-To-End System for Developing Models

Once the MIMIC-II ABP waveform data was transformed into the beat database, we built an end-to-end system for building and evaluating predictive models. The system is able to take data from the beat database, identify continuous segments free of excessive noise, extract learning samples using so called "aggregate-functions", from the segments, and train and evaluate thousands of different kinds of models. Every step of this end-to-end system is parameterized, from the details of how to extract learning samples to the types of models built.

2.4 DCAP – A System to Parallelize Tasks On the Cloud

Extracting and preprocessing data, as well as building models, are computationally expensive tasks. For this reason we built DCAP, a system to take advantage of the cloud resources at our disposal. Using DCAP we were able to run large numbers of tasks in parallel and reduce computation time from weeks to days. This system has been made open source and is described in Section 7.

2.5 Machine Learning Systems for ABP Prediction

To solve the blood pressure prediction problem we build different kinds of machine learning models. These models can be grouped into two different categories:

1. Static models (fixed lead time predictors) – these models predict blood pressure at a specific lead time into the future. We build these using Neural-Network, Discrete-Bayesian-Network with self-learned and naive structure, Naive-Bayes, Support-Vector-Machine and Decision-Tree classifiers. We train over 1000 different classifiers using DCAP to take advantage of computational resources on the cloud. Each trained classifier is a version of the above-mentioned classification algorithms with a distinct set of configuration parameters. We describe static models in detail in Section 9 and summarize their performance on the blood pressure prediction problem in Section 12.1.

2. Latent state-space models (variable lead time predictors) – these models assign a state to a patient. They do so by breaking time into "time slices" and assigning a single state to each time slice. For each time slice the latent state-space model then infers the most likely state. They can predict a patient's blood pressure at any time slice in the future by first inferring the most likely state of the patient in that future time slice and then inferring the value of the patient's blood pressure in that inferred future state. We build these models using a discrete Bayesian network with a problem-specific structure. We describe them in detail in Section 10 and provide an overview of their performance on the blood pressure prediction problem in Section 12.2.

3 Background and Related Work

3.1 Medical Record Databases

The interest in analyzing medical data has grown over the last decade. One particular focus is the analysis of data recorded in Intensive Care Units (ICU). The reason for this focus is the increasing number and complexity of the bio-medical sensors used in the ICU, which means there are large amounts of data being produced that must be quickly interpreted. As there is often a shortage of human resources, the development of intelligent patient monitoring systems that fuse data from different data streams is an active area of research [31]. In order to develop and refine such systems, large datasets of ICU data are required. Several projects have created databases containing recordings from ICUs, among them being IMPROVE [18, 26, 39] / IBIS [10, 16, 22, 38, 40], MGH Waveform DB [1], SIMON [27], Project IMPACT [15], APACHE [41], ICNARC [13], Veteran's Administration [30], and MIMIC-I and MIMIC-II databases [31, 32]. Some of these databases collect only *clinical record data*, such as medication administered; others collect only *waveform data*, that is, recordings of the patient monitoring systems such as ECG, others collect both. The databases differ in age, size, type of recordings and whether they allow open or restricted access. In addition, they differ in the type and resolution of the waveforms they collect. For a table providing a detailed comparison of the different different databases, see [32].

3.1.1 Challenges in Mining Medical Record Databases

Learning from data collected in databases such as those listed above faces several challenges. Research by the community has addressed several of these:

- *Database size* – Medical record databases often contain data from medical devices that record data at a high frequency rate. This means a large amount of fine-grained data is available. For example, the MIMIC-II waveform database is over 3 Terabytes in size because the signals were sampled at 125 hz. To address this issue Ramon et al. [29] suggest using domain knowledge to select an appropriate granularity for the data to reduce the size of the available data by aggregating it. For example, if a disease develops over hours or days, one should not search for patterns evolving within minutes or weeks.

- *Noise* – Noise is present in both clinical and waveform data. Depending on the type of data, noise takes on different forms. In waveform data the "noise" in the data is signal noise. It can arise for various reasons, for example, it may due to moving a patient or changing his height with respect to certain equipment, which may shift the measured values [29]. Another example for signal noise are sensor-specific artifacts. A discussion of these artifacts can be found in Section 3.3. Another form of "noise" can be found in clinical data. Here, the fact that different people refer to the same product using different names [29] or that people make slight mistakes when writing the name of a product can be considered noise.

- *Individuality of patients* – Different patients have different characteristics, making it difficult to compare absolute values of attributes between patients. Depending on age, height and sex, certain patients may react differently to the same medication. To address this issue Ramon et al. [29] use a "two-level Bayesian approach to learn a model that depends on patient-specific parameters". To estimate these patient-specific parameters they use a second learner. They use this two-level Bayesian approach when they perform several binary prediction tasks on a small set of data. One such example of a prediction task is predicting if a patient will still be alive N days in the future.

- *Inability to link data from multiple sources* – There is a large amount of different types of data available: demographic, historical, medication, and treatment data. Although all this data is collected, it is often collected from different sources and the records must be matched to each other in order to take advantage of related data. This is a difficult task. For example, in the case of the MIMIC databases, only a subset of the records in the MIMIC II clinical dataset have been matched to the MIMIC waveform dataset or vice versa [3].

- *Data manipulations performed to protect patient privacy* – An additional hurdle to learning from medical data and to linking data records is that certain types of data, such as dates, are considered protected health information and have been scrambled to ensure privacy [3].

3.1.2 The MIMIC-II Database

There is a significant body of research using data from medical record databases. In particular, the MIMIC-II database mentioned above is often cited as a source of data and, furthermore, it is the main source of data used in the research described in this thesis.

The Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II) research database was established by "applying automated techniques to aggregate high-resolution diagnostic and therapeutic data from a large, diverse population of adult intensive care unit patients" [32]. It is freely available and is "intended to support epidemiologic research in critical care medicine and serve as a resource to evaluate new clinical decision support and monitoring algorithms" [32]. The first release of the MIMIC-II database encompasses "virtually all adult patients admitted to ICUs at Boston's Beth Israel Deaconess Medical Center during the period 2001-2007; additional MIMIC-II data collection is ongoing". The database is currently in it's 3rd release and includes data from medical, surgical, coronary, cardiac and neonatal ICUs [2, 32].

The database is made up of two different data sets, each containing different kinds of data. The first is the clinical dataset, which includes items such as time-stamped, nurse-verified physiological measurements, diagnostic laboratory results, etc.; the second, the physiological or waveform dataset, includes recordings from monitoring systems [32]. Physiological waveforms — such as electrocardiogram, blood pressure, pulse plethysmogram, and respiration — were sampled at 125Hz, and both the raw recordings (waveform data) and trend data (numeric data) that was updated every minute were stored.

Due to the fact that "MIMIC-II Waveform and Clinical Datasets have been collected from different sources, it was not known initially which waveform and clinical records are associated with the same patient.” [3] The process of matching waveform data to clinical data is an ongoing process. The MIMIC-II Waveform Database Matched Subset “contains all MIMIC-II Waveform Database records that have been associated with MIMIC-II Clinical Database records.” [4]

3.2 Research Using Medical Record Database Data

Several studies have focused on improving the algorithms used by bedside monitoring equipment that triggers alarms. Zong et al. [43] show that it is possible to reduce false arterial blood pressure (ABP) alarms from 26.8% to 0.5% while accepting 99.8% of true ABP alarms. This is because false blood pressure alarms often stem from artifacts in the physiological ABP signal. Zong et al. designed a fuzzy logic algorithm that fuses both ABP waveform and electrocardiogram waveforms to take advantage of the fact that the two waveforms are correlated.

Li et al. [20] also take advantage of the correlation between these waveforms to design a robust heart rate estimation method. The necessity for such an algorithm is because “Physiological signals such as the electrocardiogram (ECG) and arterial blood pressure (ABP) in the intensive care unit (ICU) are often severely corrupted by noise, artifact and missing data, which lead to large errors in the estimation of the heart rate (HR) and ABP.” Their heart rate estimation algorithm fuses multiple leads by “tracking heart rate estimates” with a separate Kalman filter for each lead and then fusing the separate estimations.

Saria et al. [33,34] focused on knowledge discovery in electrocardiogram signals. They developed a time-series topic model that relied on latent variables to model heart rate and respiratory rate for premature infants in the neonatal ICU. They used supervised learning to predict the state/condition of an infant as determined by the doctors when it is released. Syed et al. [36] developed “fully automated techniques for analyzing large amounts of cardiovascular data”. Furthermore, Syed et al. [37] identified motifs in electrocardiogram data by reducing the physiological waveform to finite discrete symbols and then attempting to find patterns that may be associated with sudden cardiac death.

There has also been a focus on predicting specific acute events using machine learning and statistical tools. Ennet et al. [8] designed an algorithm to predict acute respiratory distress syndrome. Their data set included 345 'stable' and 279 'unstable' patients from the MIMIC-II clinical dataset, and they developed a rule-based prediction algorithm with a specificity/sensitivity ratio of either 80%/60% or 90%/50%.

3.2.1 Predicting Acute Hypotensive Episodes (AHE)

Recent work has focused on predicting Acute Hypotensive Episodes (AHE) using the MIMIC data, where an AHE is defined as a time period during which a patient’s blood pressure drops. This focus stemmed from the challenge hosted by Physionet in 2009. Physionet[1] hosts an annual challenge[2] “inviting participants to tackle clinically interesting problems that are either unsolved or not well solved”, and part of the 2009 challenge was to predict whether or not a patient would suffer an AHE. The challenge defined an AHE as “an interval in which at least 90% of the non-overlapping one-minute averages of the arterial blood

[1] http://www.physionet.org/

[2] http://www.physionet.org/challenge/

pressure waveform were in the acute hypotensive range during any 30 minute window within the interval". It then defines "the acute hypotensive range to include MAP measurements no greater than 60 mmHg, and [...] no less than 10 mmHg" [24]. Participants were challenged to develop "automated techniques for predicting AHEs up to an hour in advance in selected ICU patient records, using any data available before the forecast window for each record." [24] The data provided for the challenge came from the MIMIC-II database and included a training set of 60 records of which half contained an AHE event and half did not. The testing set consisted of 40 records of which 16 contained an AHE event. We provide a brief overview of the contestants:

- Mneimneh [23] et al. explored three different approaches to solving the problem: reconstructed phase-space neural networks, nearest neighbor, and a rule based approach. Their previous research had shown that "MAP itself is an excellent indicator for predicting acute hypotensive episodes". Mneimneh et al. achieved their best predictions with the rule based approach. The rule based approach achieved a sensitivity of 92.85% and a specificity of 88.46%

- Fournier et al. [9] trained a KNN classifier. To determine features they first extracted the mean and standard deviation of the Systolic, Diastolic and Mean Arterial Blood Pressure numeric records. As mentioned in Section 3.1.2, the numeric signal contains trend data of the raw waveform signal updated every minute. They used Information divergence (or Kullback-Liebler divergence) to "identify the most discriminative features". They then trained a 1-NN classifier and achieved an accuracy of 80%.

- Jousset et al. [17] explored the numeric heart rate, respiration, diastolic, systolic and mean arterial blood pressure signals. They extracted statistical parameters from these signals such as the standard deviation, skewness and the kurtosis. They then trained support vector machines using a linear kernel to select the best subset of features. The best subset contained only features from the MAP signal, and their method using only this best subset of features yielded a "limited performance" with a classification accuracy of 75%.

- Chen et al. [5] hypothesized that only basic ABP information is needed to predict an AHE. They investigated the capabilities of five different indices to detect AHE using "straightforward classification schemes" and concluded that all indices "performed well". The indices were:

  1. The "5-min average of the MAP vital signs (ABPMean) before the forecast window"
  2. The "5-min average of the MAP vital signs (ABPMean) before the forecast window"
  3. The "optimal exponentially weighted average of the 10-hr ABPMean before the forecast window"
  4. The "predicted ABPMean at the midpoint of the forecast window via linear regression of the 1-hr ABPMean before the forecast window"
  5. The "5-min average of the diastolic ABP vital signs before the forecast window"

6. A combination of features 2, and 5

- Langely [19] et al. performed a visual inspection of the MAP data and, from their observations, derived two indices. Using these indices they achieved an accuracy of 70%. The two indices they used were:

  1. "aMAP", the average MAP across the recording of the patient
  2. "AHE index", the proportion of MAP in a 30 minute sliding window falling below a threshold pressure

- Chiarugi et al. [6] derived several time series from the numerics ABP data, including heart rate, systolic ABP, mean ABP, diastolic ABP and MAP. To fill missing gaps in the time series they used linear interpolation, and to remove artifacts they employed a median filter. From these derived time series they extracted several features, including the "mean value of the systolic, mean and diastolic ABP in the last 5 hours before T_0 and the mean value of the systolic, mean and diastolic ABP in the last hour before T_0". Using these features they trained both decision tree and support vector machine classifiers. The decision tree classifiers performed better than the support vector machine classifiers and had an accuracy of 75%.

- Henriques et al. [14] built generalized regression neural network multimodels. They trained them "using arterial blood pressure signals obtained from MIMIC-II 'numerics record' dataset". They achieved an accuracy of 92%.

Separately from the Physionet 2009 challenge, Marzyeh et al. [11] also focused on predicting AHE an hour before it's occurrence. Using a variety of techniques such as Parzen models of normality, logistic regression and neural networks, they built models that achieved an accuracy of 82%.

3.3 Preprocessing Waveform Data

Often, when preprocessing ABP waveform data, it is useful to detect beat onsets. Zong et al. [42] developed an open source algorithm capable of detecting beat onsets in the ABP waveform signal. The algorithm first passes the ABP waveform signal through a low pass filter; it then converts the signal into a slope sum function signal and applies a decision rule to determine pulse onset. To evaluate their algorithm they created a manually-edited, reference ABP signal database and determined that "for 96.41% of the 39,848 beats in the reference database, the difference between the manually-edited and algorithm-determined ABP pulse onset was less than or equal to 20ms".

As mentioned in Section 3.1.1, medical record databases are often afflicted by noise. The MIMIC-II waveform database is no different, and several papers have been published on what type of noise is inherent in the database and how it can be filtered out.

Li et al. [21] discuss six different artifacts that occur in the arterial blood pressure waveform:

1. Saturation to ABP maximum, an effect that is "likely due to the flushing of the arterial line".

2. Saturation to ABP minimum, an effect that “may be due to a transient constriction in the arterial line such as pinching from arm movement”.

3. Reduced pulse pressure, an effect that “can be due to damping caused by thrombus in the arterial line”.

4. Square wave, an effect that is due to a fast flush test.

5. High frequency, an effect that may be “related to movement artifact [sic] or disturbance of the transducer (such as dragging a cloth over the arterial line)”.

6. Impulse artifact, an effect that “could be due to motion, or a sharp mechanical artifact such as crimping of the tubing”.

Two of these artifacts are depicted in Figure 1. They create mathematical models for these different artifacts and study their effect on clinical blood pressure (systolic, mean and diastolic) estimates.

Because an ABP waveform may be used for various estimations it is important to have an estimate as to how good a signal is. In order to evaluate the fidelity of an ABP waveform Sun et al. [35] propose a *Signal Abnormality Index* (SAI) to evaluate the fidelity of an ABP waveform. The algorithm uses the beat detection algorithm of Zong et al. to detect beat onsets. It then flags abnormal beats. It detects these by “intelligently setting constraints on physiologic, noise/artifact, and beat-to-beat variation values”. Asgari et al. build upon Sun’s work and propose a projection method based on singular value decomposition to validate blood pressure beats. In their evaluations, their method “achieves a true positive rate (TPR) of 99.06%, 5.43% higher than that of the SAI, and a false positive rate of 7.69%, 17.38% lower than that of SAI.”

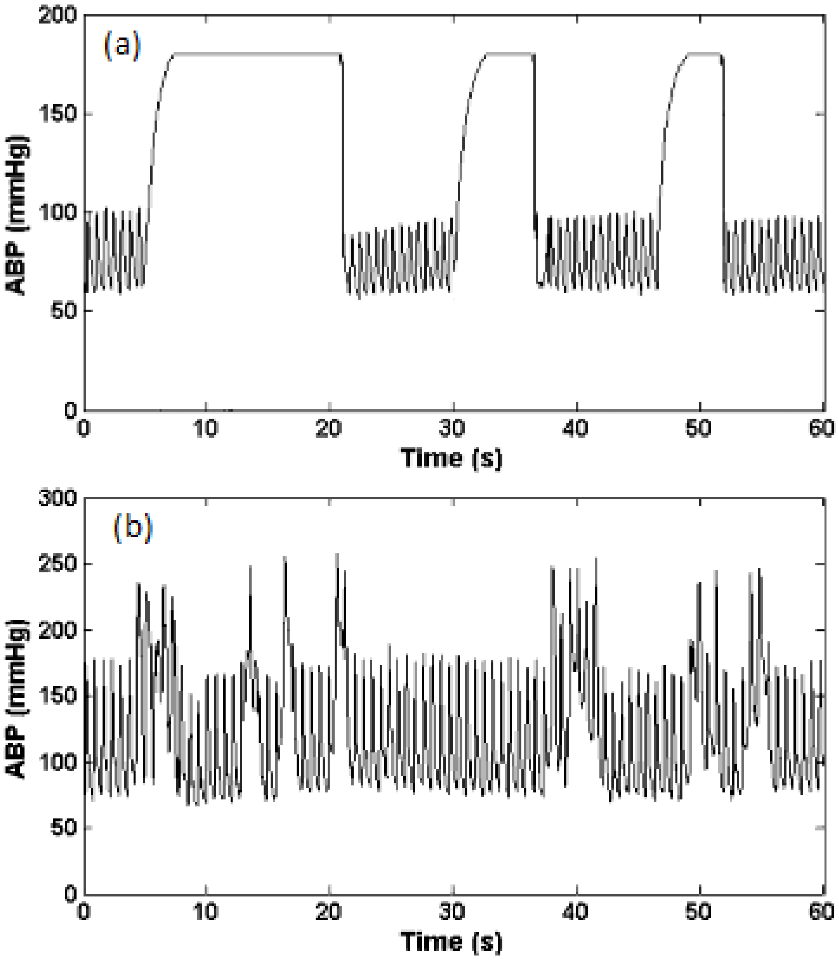


Figure 1: (a) saturation to ABP maximum and (b) impulse artifact, images from [21]

4 Problem Formulation

Let $m[n]$, where n is in the range $t_0..t_e$, be a discrete-time arterial blood pressure signal derived by sampling a continuous blood pressure signal at 125 hz, and let $m[t_i..t_j]$ be a contiguous subsequence of $m[n]$. We then define our problem statement as follows. Given a lag-time γ, a lead-time τ and a prediction-window-time α our goal is to create a model that, when provided $m[t-\gamma..t]$, makes a prediction as to what the average value will be for $m[t+\tau..t+\tau+\alpha]$. See also Figure 2. This prediction is not a continuous value in mmHG, but instead is one of 10 different ordinal classes. The following table summarises what blood pressure value each class stands for:

Class label	mmHG values represented
1	0-55
2	55-60
3	60-65
4	65-70
5	70-75
6	75-80
7	80-85
8	85-90
9	90-95
10	95-300

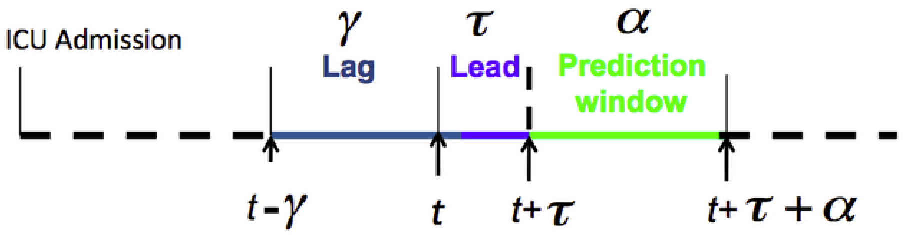


Figure 2: Lag, lead and prediction window times

5 Dataset

To create and test models that can predict blood pressure we use blood pressure data from the MIMIC-II waveform dataset. As discussed in Section 3.1.2, the MIMIC-II database contains two different datasets, physiological waveform data and clinical data, both of which were obtained from hospital information systems. These two datasets were gathered from separate data sources and only a subset of each dataset has been matched with the other. As discussed in Section 2.1, we are interested in predicting the ABP signal using only features derived from the signal itself. For this reason we use data from the entire MIMIC-II waveform dataset. Section 5.1 provides an overview of the MIMIC-II waveform dataset and it's limitations, and Section 5.2 provides an overview of how we analyze ABP waveform data.

5.1 MIMIC-II Version 3 Waveform Dataset

As of the date of this thesis the MIMIC-II waveform dataset [12] was last updated in March 2012 and has records from a total of 23,180 patients. Each record may contain one or more of a variety of different waveform recordings, including:

- ECG (electrocardiographic) waveforms: AVF, AVL, AVR, I, II, III, MCL,
- MCL1, V (unspecified precordial lead), V1, and V2
- BP (continous blood pressure) waveforms:
  - ABP: arterial blood pressure (invasive, from one of the radial arteries)
  - ART: arterial blood pressure (invasive, from the other radial artery)
  - CPP: cerebral perfusion pressure
  - CVP: central venous pressure
  - FAP: femoral artery pressure
  - ICP: intracranial pressure
  - LAP: left atrial pressure
  - PAP: pulmonary arterial pressure
  - RAP: right atrial pressure
  - UAP: uterine arterial pressure
  - UVP: uterine venous pressure
- PLETH: uncalibrated raw output of fingertip plethysmograph
- RESP: uncalibrated, respiration waveform

Most records contain only a small subset of these possible waveforms, and the waveforms that were recorded depended on choices made by the ICU staff. Most records include one or more ECG signals and often include ABP waveforms. Our focus is the ABP waveform specifically, and out of the 23,180 patient records, 6,232 patient records have a blood pressure waveform recording. Although this number of patients may not seem like "big data", two factors make the data available large. First, for each patient, their ABP was recorded anywhere from several hours to several days. Second, their ABP was recorded using a sampling rate of 125 hz. Because there is over 240,000 hours worth of ABP data in the waveform database, over 2 Terabytes worth of uncompressed ABP data is available.

5.1.1 Organisation and Limitations of the Database

The MIMIC-II waveform dataset is organised in the following manner[3]. For each recording there is a single record directory containing both a waveform record and a numerics record. Our research uses only the waveform record. The waveform record is broken up into multiple segments, where each segment contains an uninterrupted recording and the signal gains do not change at any time during the segment.

Certain limitations must be taken into account when reading data from these waveform records:

1. *Gaps in the records*: As records in the database can be very long, for example, a single patient may be monitored for several days or even weeks, the physiological signals often have interruptions. These gaps occur because monitors may be disconnected occasionally for a varying amount of time.

2. *Patient identification*: The waveform data was extracted from raw dumps collected from bedside monitors. A monitoring session usually starts when a patient is admitted and ends when the patient is discharged. However, monitors are not always reset when patients are discharged. In this case a given record may include data from two or more patients. Usually, when this happens there is a gap that is typically of an hour or more in duration. For this reason all records with gaps of one or more hours have been split into multiple records.

5.2 Exploring Arterial Blood Pressure Signals

A record's ABP waveform can be accessed either indirectly via the PhysioBank ATM website[4] or directly from the database[5] using the wfdb software package[6]. Once downloaded, the waveform signals still need to be adjusted by subtracting the base and dividing by the gain. An example ABP waveform belonging to record 3218321 is depicted in Figures 3 and 4. Figure 3 shows the first 22 hours of record 3218321's ABP signal. In this figure one can see examples of the two types of artifacts described by Li et al. [21] and mentioned in Section 3.3. Figure 4 shows the first 20 seconds of record 3218321's ABP signal. At this level of granularity it is possible to visually identify individual blood pressure beats. We note two phenomena that are visible in Figure 4

1. At around 10 seconds into the recording there appears to be a disturbance in the signal.

[3] http://www.physionet.org/physiobank/database/mimic2wdb/

[4] http://www.physionet.org/cgi-bin/atm/ATM

[5] http://www.physionet.org/physiobank/database/mimic2wdb/31/

[6] http://www.physionet.org/physiotools/wfdb.shtml

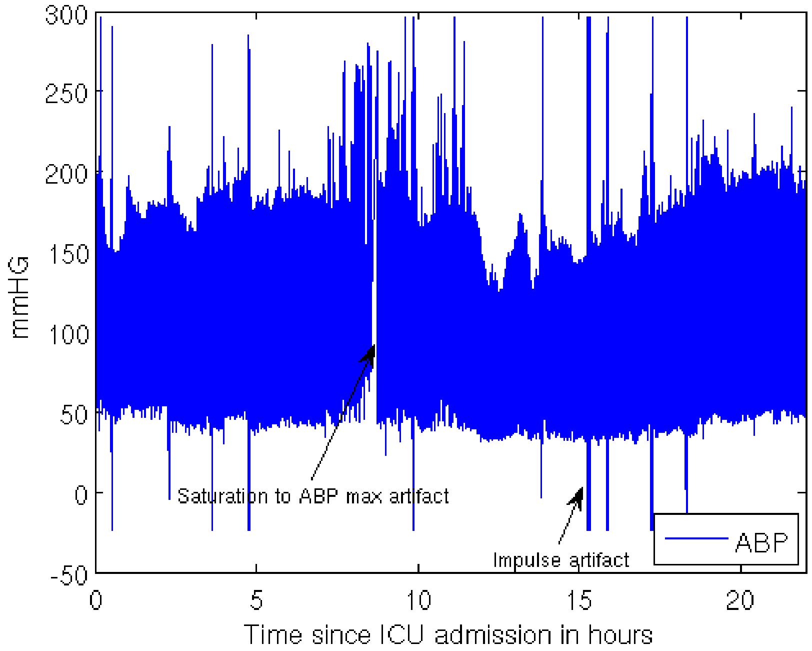


Figure 3: ABP waveform of record 3218321

2. At the beginning of the signal and at around 17 seconds into the signal there appears to be a blood pressure beat that is diminished in comparison to the majority of the beats.

As neither of these phenomenon are described in the literature we surveyed, we asked physicians David Dernoncourt and François De Forges what they thought of them. They replied that the disturbance around 10 seconds was certainly due to noise, and the phenomenon at the beginning of the signal and at around 17 seconds into the signal were most likely extrasystoles. Citing perez et al. [28], "Kennedy et al. demonstrated that frequent (>60/h or 1/min) and complex PVCs could occur in apparently healthy subjects, with an estimated prevalence of 1-4% of the general population", they explained that extrasystoles are a much more common phenomena in acute inflammatory conditions, and that they can be triggered by ICU procedures, such as intra-auricular catheterism.

5.2.1 Beat Onsets

The ABP signal is an oscillatory waveform that repeats with a period known as the *beat duration*. This period varies not only from patient to patient, but also within an individual patient's signal. In a single beat the blood pressure rises from a low value, called the *diastolic pressure* P_d, to a peak value called the *systolic pressure* P_s; this phase of the signal is called the *anacrotic limb*. After the anacrotic limb the signal declines and then has a bump (a small rise and fall) called the *dicrotic notch*. After this bump the blood pressure drops back down to P_d. This second phase of the the beat is called the *dicrotic limb*.

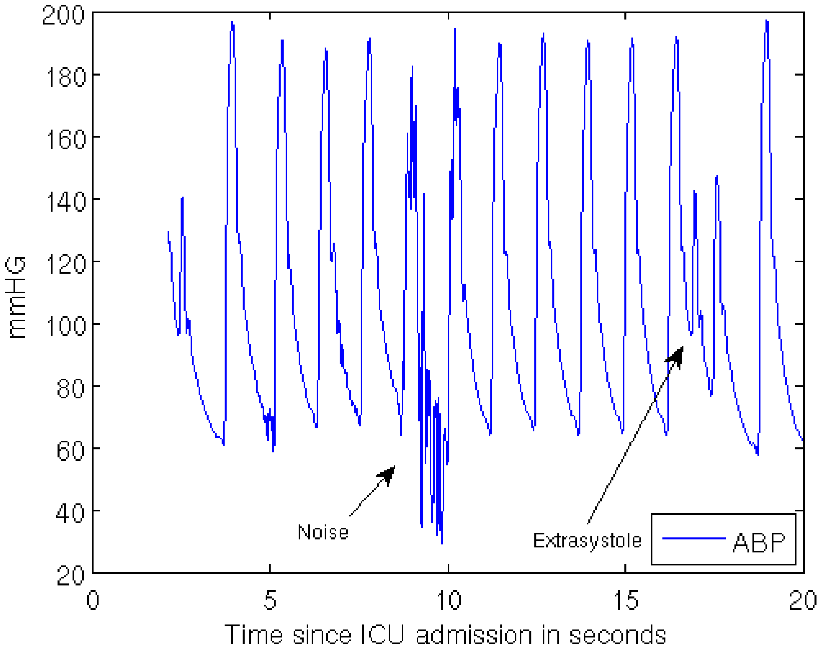


Figure 4: Close up of record 3218321's ABP waveform

Image 5 provides an overview of an individual blood pressure beat.

To detect beat onsets, we applied the algorithm developed by Zong et al. [42] and mentioned in Section 3.2 to the MIMIC ABP blood pressure records. Their beat detection algorithm is part of the WFDB toolbox[7]. Figure 6 shows the result of applying the algorithm to the first 20 seconds of record 3218321's ABP waveform. The algorithm detects and marks every real beat onset correctly. However, it marks several points that are clearly not beat onsets. This indicates that both noise and extrasystoles increase the algorithm's rate of false positives.

To analyze the individual beats within an ABP signal we need to detect which parts of the signal are noisy and which beats have been corrupted. To remove noisy and corrupt beats we apply the signal abnormality index developed by Sun et al [35]., which determines if a beat as abnormal or not. The algorithm that calculates the signal abnormality index is very quick to implement and run. It flags a beat as abnormal if the beat fulfills any of the following criteria in List:

- Systolic pulse pressure greater than 300mmHg
- Diastolic pulse pressure less than 20mmHg
- Mean arterial pressure less than 30mmHg or greater than 200 mmHg
- Heart rate less than 20 bpm or greater than 200 bpm

[7] `http://www.physionet.org/physiotools/wag/wabp-1.htm`

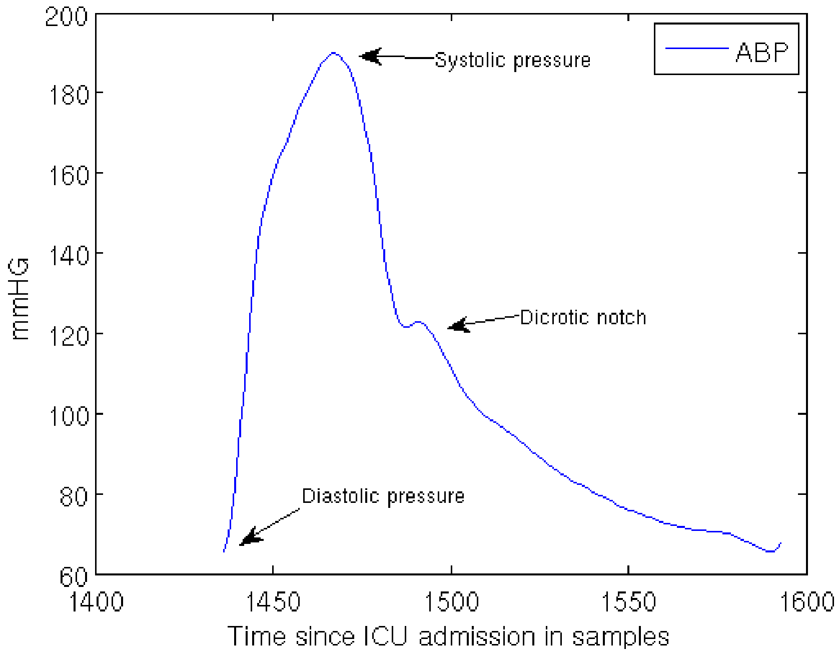


Figure 5: A single ABP beat

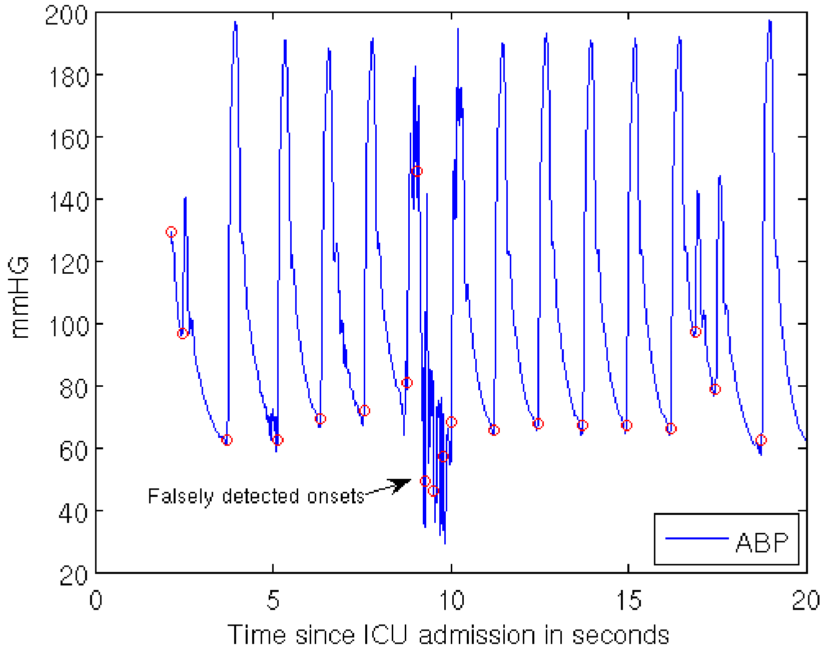


Figure 6: Detected beat onsets, marked with red circles

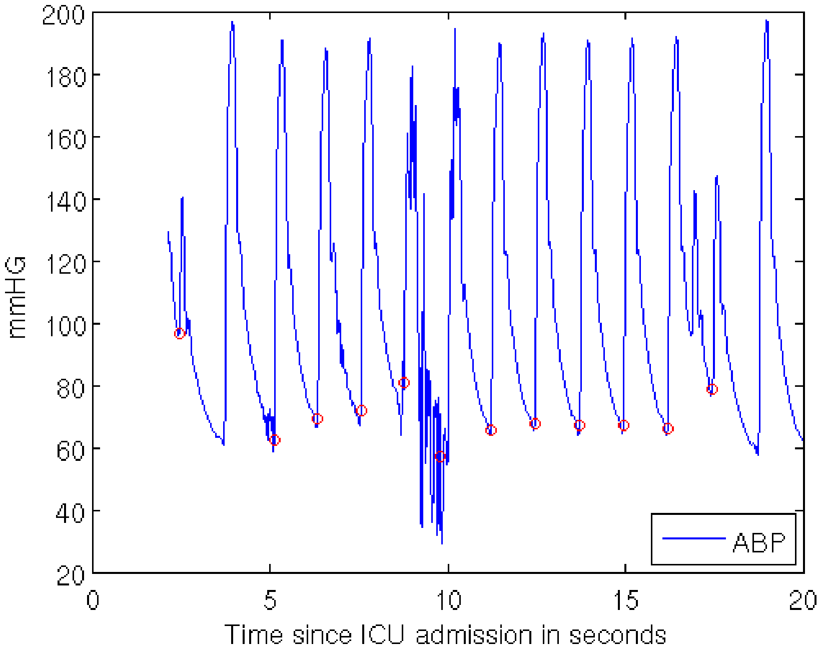


Figure 7: Onsets belonging to beats flagged as "abnormal" removed

- Pulse pressure less than 20 mmHg
- The mean negative slope is less than -40 mmHg / 100ms (noise detection)
- The difference in the systolic blood pressure between the beat and the previous beat is greater than 20 mmHg
- The difference in the diastolic blood pressure between the beat and the previous beat is greater than 20 mmHg
- The difference in the duration between the beat and the previous beat is greater than $\frac{2}{3}$ of a second.

The effect of applying the signal abnormality index to the ABP signal in Figure 6 is demonstrated in Figure 7, where the first 20 seconds of record 3218321's ABP waveform have been plotted along with the beat onsets that the signal abnormality index flagged as normal. As one can see, most of the beat onsets belonging to noisy beats have been removed.

6 Toward Creating a Beat Feature Database

Our goal is to analyze all the ABP signals contained in the MIMIC-II waveform dataset at the individual beat level. We understand analyzing an ABP signal at beat level as the following. First, we identify the contiguous segments of the signal that belong to each beat, and then we extract various signal features from these contiguous segments. We refer to these extracted signal features as *beat features*. To make these features readily accessible we use a pipeline to extract all beat features from the MIMIC-II waveform dataset and store them in a database. Each pipeline step is described in more detail below. The steps are:

1. Raw ABP signal storage
2. Preprocessing ABP waveforms
3. Beat onset detection, beat validation and gap discovery
4. Beat feature extraction

6.1 Raw ABP Signal Storage

As access to the remote MIMIC-II database is slow, our first task was to set up a mirror of the MIMIC-II database containing only the MIMIC-II ABP waveform dataset. We downloaded all records containing an ABP waveform, adjusted the waveform for base and gain, and then zipped and stored each record's adjusted ABP waveform data-files in individual folders on an ftp server. We refer to such a file containing a record's ABP waveform adjusted for both base and gain as a raw ABP waveform file. Each raw ABP waveform file is a two column, comma-separated value (CSV) file of samples, where the first column gives a sample's index and the second the sample's signal value. As an example of such a file is given in the following table, which contains the first few rows of 3218321's raw ABP waveform file, starting with sample 273.

Sample	Signal Value
273	129.104526
274	129.104526
275	126.525562

6.2 Preprocessing ABP Waveforms

As a preparation for the next step in the pipeline, beat onset detection, we envision using sophisticated filters to remove noise while keeping the high level frequency information contained in a beat. By removing noise before applying the beat onset detection algorithm it should be possible to reduce the false positive and false negative rate of the beat onset detection algorithm. However, developing and implementing such filters was not part of this thesis and at the completion time of this thesis, this step has not yet been implemented. It will be implemented in future work. The output of this pipeline step is the preprocessed ABP waveform files, which are CSV files in the same format as the raw ABP waveform files.

6.3 Beat Onset Detection, Validation and Gap Discovery

In this step each ABP waveform is passed through the beat onset detection algorithm designed by Zong et al. [42] . The output of this beat onset detection algorithm is a list of time points specifying when each onset occurs. We define a beat to be the from the beginning of an onset to the start of the next onset.

Once the beats within an ABP waveform have been detected, we classify each beat using a modified version of the signal abnormality index defined by Sun et al. [35] . Using the same criteria as Sun et al., we determine the signal abnormality index of each beat and assign it a value between zero and two. If the signal abnormality index declares a beat as normal we mark it as valid (1). If the signal abnormality index declares a beat as abnormal we mark it as invalid (0). We also introduce an additional criteria for the case when a beat's duration is greater than 750 samples, a duration that corresponds to 6 seconds. In this case we mark the 'beat' as a gap (2) . A gap indicates a disturbance in the signal that is significant enough that the data prior and posterior to the gap should not be considered as belonging to the same contiguous segment.

The result of this pipeline step is another CSV file called the *record beat file*. This file contains three columns. The first column is the beat onset, containing the sample when the beat starts; the second is the beat end, containing the sample when the beat ends; the third is the validity index, an index indicating if the beat is valid (1), invalid (0) or if the beat represents a gap (2). As an example of such a file is shown in the table below. The table contains the first few rows of 3218321's record beat file.

Beat onset (in samples)	Beat end (in samples)	Beat end (in samples)
273	314	0
274	476	1
275	655	0

6.4 Beat Feature Extraction

The final step of the pipeline is to extract features from each beat. Let $m[b_o..b_e]$, be a contiguous segment of $m[n]$ where b_o is the beat onset of any beat and b_e is the end of that beat. We then refer to a function

$$s = f(m[b_o..b_e]) \tag{1}$$

that operates on a contiguous segment and produces a scalar value as a *beat-feature function*, and the scalar result as a *beat feature*. One such beat-feature function is the mean ABP of a beat, and the function that calculates it is:

$$b_{map} = \frac{1}{b_e - b_o} * \sum_{i=b_o}^{b_e} (m[i]) \tag{2}$$

For each beat feature we have a beat-feature file. This is a CSV file with one column. Each value is the scalar value result of the beat feature function. Every beat feature file belonging to a specific record has the same number of entries as that record's beat file. Scalar values belonging to beats flagged as invalid or as a gap are set to not-a-number (NaN).

There are many different kinds of beat features that can be extracted, from time domain features such as the mean, to features that analyze the frequency spectrum of each beat. Exploring and developing beat features for the beat database was not part of this thesis, and at the time of completion of this thesis, only the b_{map} beat feature function was implemented. The implementation of additional beat feature functions is future work.

6.5 Database File Structure

We use a simple scheme to store raw ABP waveform files, beat files and beat feature files. At the top level we have three directories, each corresponding to one of the types of files. Each top-level directory has a subdirectory corresponding to each MIMIC waveform record. Each of these subdirectories contains the files corresponding to both the top level directory as well as the waveform record. The following list depicts the file structure:

- rawABPwaveforms
  - 3218321
    - record.csv
  - ...
- recordBeatFiles
  - 3218321
    - recordBeatFile.csv
  - ...
- beatFeatures
  - 3218321
    - b_{map}.csv
    - ...
  - ...

Once the preprocessing step has been implemented another top-level directory will be added, and all record beat files as well as all beat features will be recalculated. Any additional beat features implemented in future work will be stored under their corresponding subdirectory in the beatFeatures top-level directory.

6.6 Overview of the Dataset in the Beat Database

In the table below we provide several summary statistics of the beat database we created. Figure 8 shows the distribution of the ordinal blood pressure classes defined in Section 4 within the beat database, if we assign each beat to its corresponding ordinal class using the beat's b_{map} feature.

Total number of records with ABP waveforms	6.187×10^3
Total number of beats	1.2885×10^9
Total number of valid beats	1.0921×10^9

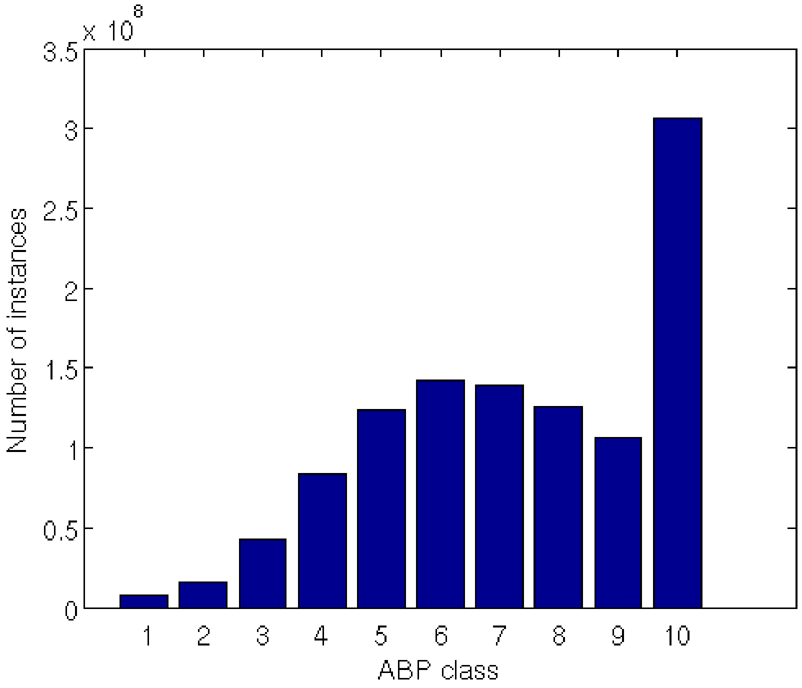


Figure 8: Class distribution of the entire waveform dataset

7 DCAP – Distributing Tasks to the Cloud

In solving the blood pressure prediction problem we encountered many tasks that are computationally very expensive, such as calculating the signal abnormality index for each beat and training classifiers for our models. As we were not able to compute these tasks in a reasonable time using a single machine we designed a system that distributes a set of tasks to machines running on the cloud (often referred to as nodes in the cloud). The system needed to fulfill the following three requirements:

1. Generic tasks - We needed to be able to run different kinds of programs on nodes in the cloud. Some of these programs were written in Matlab, others in Python.

2. Robustness - As the cloud environment is fickle, the system must recover from failure of any node without losing progress in the computation.

3. Elasticity - The availability of computation resources fluctuates on the cloud and we need to add or remove nodes from the computational task at any time.

The system we built to fulfill these requirements is called DCAP, which is an acronym that stands for "A Distributed Computation Architecture in Python". DCAP is a client-server architecture. When running, the server keeps a list of computational tasks that it will distribute to its clients.

DCAP is robust because it allows for client as well as server failure. If a client failure occurs, DCAP will schedule any tasks assigned to that client to other clients. If the server fails, DCAP is able to resume computation from the time the failure occurred as long as the server was backed up. DCAP is elastic; if computation is consuming too many resources, it is possible to terminate running client nodes. Similarly, if additional computation resources become available, it is possible to start up additional client nodes and have them connect to the server.

Two examples in which we used DCAP are the calculation of the beat abnormality index (see Section 6.3) together with the b_{map} feature extraction (see Section 6.4) and the calculation of static models (see Section 9.2.1). In the first example we calculated the abnormality index and extracted the b_{map} feature for more than one billion beats using 50 nodes, and the computation took one and a half days. In the second example we built more than one thousand static models using 80 nodes, and the computation took 2 days.

We deployed DCAP on the OpenStack[8] based cloud environment run at the Computer Software and Artificial Intelligence Laboratory at MIT[9]. We have made DCAP's code[10] along with its documentation available online[11].

[8] http://www.openstack.org/

[9] http://tig.csail.mit.edu/wiki/TIG/OpenStack

[10] https://github.com/byterial/dcap

[11] http://byterial.blogspot.com/2013/02/dcap-distributed-computation.html

8 Overview of Building and Evaluating Models

In Section 2.5 we described how we built and evaluated two different types of models to predict blood pressure:

1. Static models (fixed lead time predictors) — these models predict blood pressure at a specific lead time into the future. We build these using Neural-Network, Discrete-Bayesian-Network with self-learned and naive structure, Naive-Bayes, Support-Vector-Machine and Decision-Tree classifiers.
2. Latent State-Space models (variable lead time predictors) — these models predict blood pressure at different lead times into the future. They are built using a Discrete-Bayesian-Network with a problem-specific structure.

Regardless of what types of models we build and evaluate there are several steps that are common. To begin with, building a model requires training data and evaluating requires testing data. The selection of training and testing data is important for the success of a model. We discuss the criteria for selecting training and testing data in Section 8.1, and the data we selected in Section 8.2. Training and testing data is relatively low-level data and, by itself, is not directly usable for building and evaluating models. This data must be transformed into learning and testing samples. Although the exact details for transforming the selected data depend on the specific model being used, there are certain aspects that are common, in particular, the use of aggregate functions to calculate aggregate features. These are described in Section 8.3.

Having described the common steps in this section we then look at the two types of models. We first describe static models in Section 9 followed by latent state-space models in Section 10. How we evaluate these models is described in Section 11.

8.1 Criteria For Training and Testing Datasets

When building and evaluating models, an important decision that must be made is what data will be used to build the model and what data will be used to evaluate the model. The data that is used to build the model is called the *training set*. A good training set allows the model to separate data belonging to one class from data belonging to another without overfitting the model. If the training set contains significantly more samples of one class than of another it is likely that, in the feature domain, some samples from the more numerous class will overlap samples from the less numerous class. This can be due to noise or the presence of outliers in the more numerous class see Figure 9. This makes it difficult for the model to learn the difference between the two classes, and it is likely that the model will be biased towards the more numerous class. To overcome this, four different approaches can be used:

1. Develop new features that allow the classes to be distinguished.
2. If the model is a classifier that is trained using an optimization function, increase the cost of misclassifying one of the less numerous classes

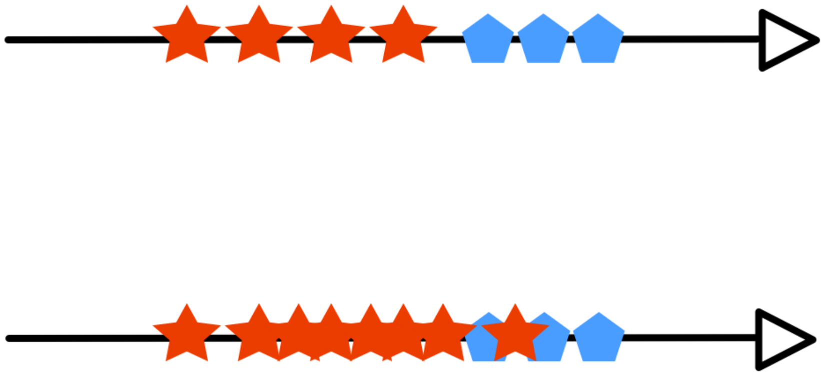


Figure 9: Unbiased and biased datasets

3. If the model is a probabilistic model that calculates a probability for each class when it makes a prediction, multiply these probabilities by weights, where the weights of the less numerous classes are greater than the others.
4. Subsample a different training dataset in which there is a more even representation of each of the classes.

The data that is used to evaluate the data is called the *test set*. For the evaluation to be meaningful the test set should fulfill the following criteria:

- The test set contains data that the model has not seen before, and the data in the test set is not correlated with the data in the training set.
- The data in the test set is representative of potential new data that the model would encounter if it were deployed.

8.2 The Subset of Data Used for Training and Testing

Looking at Figure 8 one immediately recognizes that certain blood pressure classes are far more frequent than others. In particular, the high blood pressure classes — eight, nine and ten — as well as the average blood pressure classes — four, five six and seven — are more strongly represented than the low blood pressure classes. Due to the reasons mentioned in Section 8.1, this means that models built using a training set composed of data sampled at random from the entire ABP waveform dataset may have difficulty in correctly classifying low blood pressure classes. In addition, because only one beat feature has been implemented so far, b_{map}, the current state of the beat database will exacerbate this problem. However, as described in Section 1.1, it is the low blood pressure classes that are particularly dangerous to a patient; therefore, we would like to develop models that are good at predicting the low blood pressure classes. To

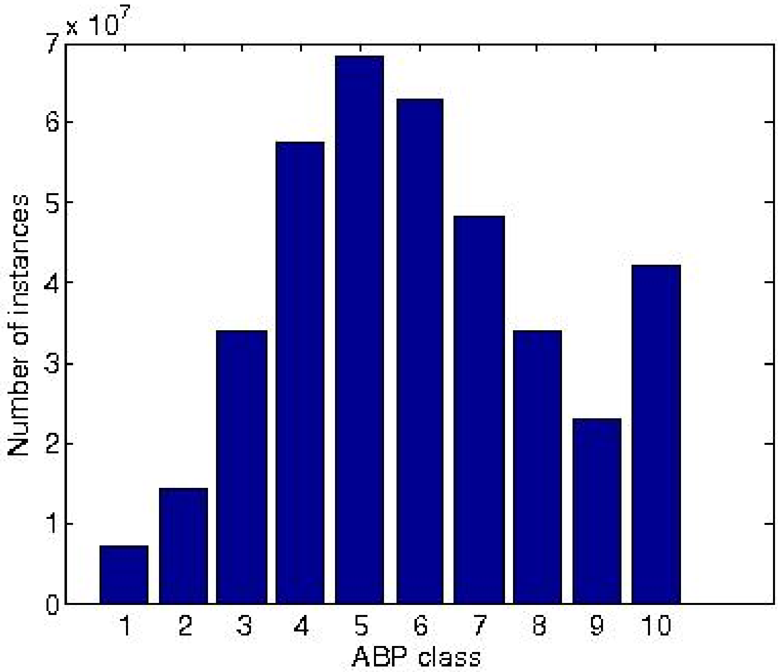


Figure 10: Class distribution of the entire waveform dataset

address this issue we subsample the entire ABP waveform data set and select the 1000 records that have the greatest number of beats in the low ABP classes. We call this subset of records the 1000-record-subset. Figure 10 depicts the distribution of the ABP classes within this record group. From this figure we see that the low blood pressure classes make up a greater portion of the distribution. To improve the performance of the models on the low blood pressure classes we use only data from the 1000-record-subset to build the training set.

When creating a testing set it is arguable if we should use data from the entire dataset or data from the subsampled data. One one hand, data from the entire dataset is more likely to represent a random new subject; on the other hand, it does not provide as good an evaluation of our models on the low blood pressure classes. For this thesis we decided to put the emphasis on predicting low blood pressure classes, and we use only data from the 1000-record-subset to build the testing set.

8.3 Data Transformation and Aggregate Functions

Although the exact data transformation used is specific to the type of model, static or latent state space, there is one aspect that they have in common; both use aggregate functions to create samples.

8.3.1 Aggregate Functions and Features

The beat-feature data stored in the beat database is at a fine level of granularity and is not directly suitable for training/testing models. A subsequence of beat features from the database can be considered to be a new time series that has been sampled at uneven intervals. We call this new time series a *beat-sample time series*, and we refer to the beat-sample time series of a specific beat feature as *⟨beat-feature-name⟩ time series.* For example, the time series of the b_{map} feature is called the b_{map} time series. The reason the intervals are uneven is because the beat-period varies from one beat to the next. If we move a sliding window of time-duration t_s across a beat-sample time series from the database, we can apply a function to this window to extract "higher-level" signal features from this time series. We call the applied function an *aggregate function* and a signal feature extracted from a beat-sample time series using such a function an *aggregate feature.* Let $b[n]$ denote a beat-sample time series of length L, that is, n is in the range $0..L-1$. Also, given a sequence $s[i]$ for i in $0..m$, let $Fd(s)$ denote a new sequence with $Fd[j] = s[j+1] - s[j]$ for j in $0..m-1$, that is, $Fd(s)$ is the sequence of first-order forward differences of s. We then define the following aggregate functions:

- Aggregate-Mean: $m_t = \frac{1}{L}\sum_{i=0}^{L-1} b[i]$
- Aggregate-Standard-Deviation: $sd_t = \sqrt{\frac{1}{L}\sum_{i=0}^{L-1}(b[i]-m_t)^2}$
- Aggregate-Skew: $sk_t = \frac{\frac{1}{L}\sum_{i=0}^{L-1}(b[i]-m_t)^3}{sd_t^3}$
- Aggregate-Kurtosis: $ku_t = \frac{\frac{1}{L}\sum_{i=0}^{L-1}(b[i]-m_t)^4}{sd_t^4}$
- Aggregate-Trend: $tr_t = \frac{Covariance(i,b[i])}{sd_t^2} = \frac{\frac{1}{L}\sum_{i=0}^{L-1}(i-\frac{L}{2})(b[i]-m_t)}{sd_t^2}$
- Aggregate-Median-Velocity: $v_t = Median(Fd(b[n]))$
- Aggregate-Median-Acceleration: $a_t = Median(Fd(Fd(b[n])))$

Note that, when applying an aggregate function, we have to omit any invalid beat-features. These are recognized by a value of NaN, as described in Section 6.4. Note also that it may be possible to apply some, or all, of these functions to different kinds of beat features, although for this thesis there is only one beat feature, b_{map}, to which we can apply all of the above functions.

With respect to using aggregate functions, the primary difference between the two model types is the selection of the subsequence and the choice of window when extracting aggregate features.

9 Static Models

The first kind of blood-pressure prediction models we describe are static models. These models do not model the time aspect of the ABP signal but, instead, are built by learning from a dataset of learning samples made up of feature vectors and class labels. The components of the feature vectors are the aggregate features. To create the dataset of learning samples for these models we must transform the data into feature vectors; this process is described in Section 9.1. Afterwards, in Section 9.2, we describe a framework we built that can build many kinds of static models based on common classifier algorithms.

9.1 Transformation of Data Into Learning Samples

To transform the data we recall the blood pressure prediction problem described in Section 4. Our goal is to extract features from a section of the signal called the lag window, which is γ minutes long, and then use these features to predict the blood-pressure class in the future. In particular, we wish to predict the blood-pressure class that lies lead time τ in the future. Furthermore, the value of this class is to be calculated for a prediction window of duration α minutes starting at lead time τ.

To calculate the learning samples used to train a model we consider a sequence of beat features as a new beat-feature time series, as described in Section 8.3.1. Using the beat-feature files containing validated beats, as described in Section 6.3, we first extract contiguous sequences, that is, sequences that do not include a gap indicated by a 2 in the record-beat file. We then transform each contiguous sequence into a set of aggregate features for the learning sample. We do this by sliding a window of γ minutes across each contiguous sequence. The sliding window moves in steps of γ minutes, so as not to overlap with the previous position of the sliding window. At each step we extract the aggregate features from the time series within the sliding window by applying the aggregate functions as described in Section 8.3.1. These aggregate features form the feature vector of one learning sample. We stop moving the sliding window as soon as the end of the sliding window is $\tau + \alpha$ minutes away from the last value of the contiguous time series. This leaves enough samples for the lead time and prediction window corresponding to the last feature vector in the contiguous sequence.

To help understand this process we give an example of extracting learning samples from a contiguous b_{map} time feature series. To make the example simple we will assume a short duration of the b_{map} time feature series of 10 seconds. Again, to simplify the example we choose the values of γ, τ and α to be short, namely 3, 2 and 2 seconds, respectively. Furthermore, again to keep the example simple, we only extract the mean aggregate feature as well as the standard deviation aggregate feature.

Let us assume the values of the b_{map} time series are as given in Table 1. We detail the steps taken to extract the learning samples.

First position the sliding window at the beginning of the time series. It's width is the lag duration, three seconds. This means it covers beats one, two and three. We extract the mean and standard-deviation aggregate features:

Mean	71.67	mmHg
Std	7.63	mmHg

Beat index	Time in seconds	b_{map} in mmHg
1	0	70
2	1	65
3	2.5	80
4	3.1	74
5	4.3	68
6	4.6	65
7	5.4	50
8	6.1	60
9	7.2	63
10	8.1	70
11	9.3	80

Table 1: An example of a b_{map} time series

To determine the associated class for this learning sample, we look lead time of two seconds into the future from the end of the sliding window and place a prediction window of two seconds. This prediction window will cover beats eight and nine. From these beats we extract the mean aggregate feature: 61.5. This means the associated class label for this learning sample is three because it is in the range 60-65 mmHg.

We then move the sliding window by three seconds so that it does not overlap its previous position. This time it covers beats four, five, six and seven, and again we extract aggregate features:

Mean	64.25	mmHg
Std	10.21	mmHg

The corresponding prediction window covers beats ten and eleven. As the mean b_{map} for these is 75 mmHg, the corresponding class label is six.

After this we move the sliding window to cover beats eight, nine and ten. However, as the prediction window would lie beyond the boundary of the contiguous b_{map} time series, the algorithm halts. In total we extract two learning samples from this time series:

Aggregate Mean	Aggregate Std	Label
71.67	7.63	3
64.25	10.21	6

9.2 Building and Evaluating the Static Models

Because the learning samples have the form of a feature vector associated with a class label, we can build and evaluate predictor models using common machine learning algorithms such as neural networks or support vector machines. Many machine learning algorithms have matured to the point where state-of-art implementations of these algorithms are readily available. However, which machine learning algorithm yields the greatest performance is not always clear. Even after deciding on a machine learning algorithm, many algorithms have a suite of parameters that need to be fine tuned to improve the algorithm's performance.

To handle this issue, we decided to take advantage of the recent technological advances that have made computation power available in the form of compute on demand. Using a client-server architecture, we built a massively parallelized, Multi-Algorithm, Multi-Parameter Suite that solves our machine learning problem with a variety of machine learning algorithms—support vector machines, neural networks, discrete bayesian networks, naive bayes, decision trees—and varies the algorithms' parameters to find not only the best performing algorithm, but also the best selection of parameters for it.

9.2.1 Multi-Algorithm Multi-Parameter Suite Architecture

The system's software architecture consists of two parts, the server and one or more clients, all running DCAP. As described in Section 7, DCAP is an architecture that was designed to distribute generic tasks to clients. For our Multi-Algorithm, Multi-Parameter Suite, these tasks consists of training a specific classifier algorithm with a specific set of parameters. We refer to such a specific task as a *classifier instance*. For example, a classifier instance could specify an SVM classifier, to be trained with a specific kernel, such as a 'polynomial' kernel, with a specific method of solving the hyperplane, such as sequential minimal optimization, along with whether or not autoscaling should be turned on.

The server stores a list of classifier instances to be trained including their parameters covering the parameter space of all the different machine learning algorithms specified at the beginning of Section 8. To train these classifier instances the server farms the tasks out to it's clients.

The process of farming classifier instance tasks to clients through DCAP is as follows:

1. The server maintains a list of untrained classifier instances.

2. A client connects to the server and requests a task.

3. The server selects the next untrained classifier instance and sends it's parameterization through the network to the client.

4. The client has a copy of the training data set locally and uses it to train the classifier instance it received from the server.

5. The client then transfers the completed model along with it's evaluation to the server and requests another task.

This process is depicted in Figure 11.

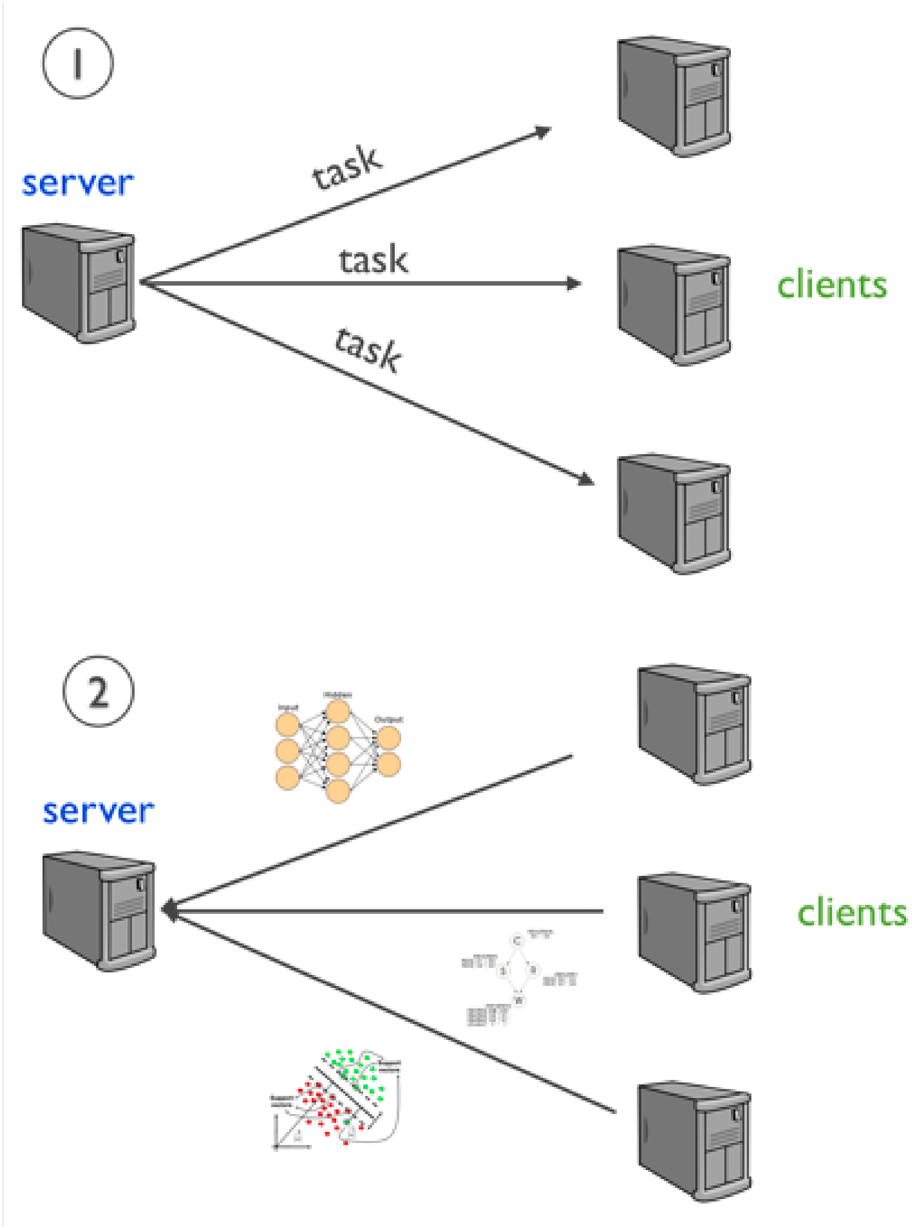


Figure 11: Training classifiers on the cloud using dcap

10 Latent State-Space Models

10.1 Modeling ABP as a Latent State-Space Model

In contrast to the models described in Section 9, latent state-space models take into account the temporal nature of an ABP signal. They do this by assuming that, because an ABP waveform belongs to an individual patient, the observed blood pressure is the result of his state. As time progresses his state may change, and such a change may result in a change in his ABP. While the patient's state is not directly observable, his ABP waveform and any features derived from it, are. For this reason the patient's state is called a *latent state*, and the observable properties, such as his ABP, are called *observables*.

One way to design such models is to model a patient's latent state not as a state that is continuously developing, but as a state that takes on only discrete values in the set $\{1, 2, ..., h\}$ at discrete points in time. To do this we divide time into slices, so-called *time slices*. For the duration of a time slice we assume that the patient's state does not change. At the end of each time slice the patient transfers to the next time slice, thereby making a state transition. For each time slice t we model the patient's state as a discrete random variable with a support of h, meaning that it takes values in the range from 1 to h. This means that if we observe a patient over T time slices, we have T random variables H_1, H_2,...,H_T that form a chain. When the patient makes a state transition at the end of time slice t, the random variable $H_{(t+1)}$ that represents the new state will be different from H_t. We model the probability distribution of the random variable H_t as a probability distribution that is conditionally dependent only on the random variable that models the previous state. That is, we assume that the successor state of a patient depends only on that patient's current state. By making this assumption we have modeled the process of a patient changing states as a Markov process, since it fulfills the Markov property, which states that the conditional probability distribution of future states depends only upon the present state.

We now give the conditional probability distributions for the state random variables. For any state random variable H_t, for $t > 1$, it's conditional probability distribution is given by the following Conditional Probability Table (CPT), which we have written out for H_{t+1}:

H_{t+1}/H_t	1	...	h
1	$P(H_{t+1} = 1 \vert H_t = 1)$	...	$P(H_{t+1} = 1 \vert H_t = h)$
2	$P(H_{t+1} = 2 \vert H_t = 1)$	...	$P(H_{t+1} = 2 \vert H_t = h)$
...	...	...	...
h	$P(H_{t+1} = h \vert H_t = 1)$	...	$P(H_{t+1} = h \vert H_t = h)$

The question remains as to what the probability distribution of the first variable in the chain, H_1, is. The answer is that it is just the prior distribution of the states.

The values that the random variables representing the patient's state take are not observable. What is observable is the state's effect on the patient's blood pressure. For each time slice at time t we model the observable effect of the state as a set of discrete random variables: $\{V_{1,t}, V_{2,t}, ..., V_{n,t}\}$. By observing the blood pressure we can determine the values these discrete random variables have taken. How we extract the effect of a patient's state on the blood pressure from

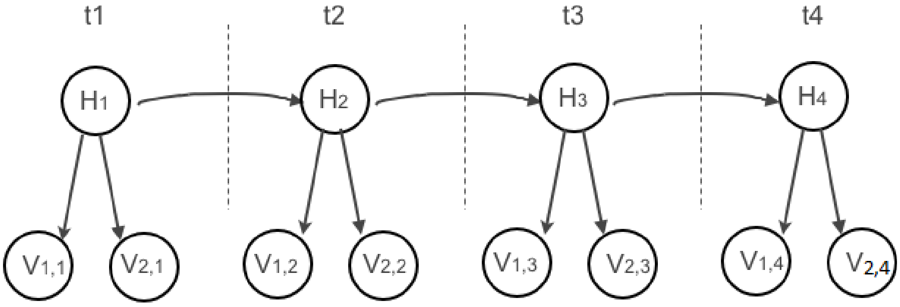


Figure 12: Latent state space model

an ABP signal and discretize it to determine the values the random variables have taken is described in Section 9.1. Because these observations depend only on the current state of the patient, the probability distribution of each observable discrete random variable with a support of p is:

$V_{i,t}/H_t$	1	...	h
1	$P(V_{i,t} = 1 \| H_t = 1)$	...	$P(V_{i,t} = 1 \| H_t = h)$
2	$P(V_{i,t} = 2 \| H_t = 1)$	...	$P(V_{i,t} = 2 \| H_t = h)$
...	...	...	...
p	$P(V_{i,t} = p \| H_t = 1)$	...	$P(V_{i,t} = p \| H_t = h)$

Figure 12 depicts the graph that models the relationship between the random variables that represent the first few time slices of an ABP waveform. Each node in the graph corresponds to a random variable, and the arrows represent conditional probabilistic relationships. Such a graph is called a discrete bayesian network. By using a discrete bayesian network to model our latent state-space model we can implement it using the open source Bayes Net Toolbox[12].

10.2 Latent State-Space Model Learning

10.2.1 Learning When the Number of States Is Known

If we assume we know the support of the state random variables, then it is possible to fit the values of the conditional probability tables, i.e., the parameters, to the training data such that the likelihood of observing the data given the parameters is maximised. The most common approach is to use an expectation maximisation algorithm. Expectation maximisation is an iterative process. To initialize the process the conditional probability distributions of the random variables are initialised to random values. The process then iterates the following two steps:

1. The expectation step — calculate the log-likelihood of the data given the model's parameters.

2. The maximisation step — compute the new parameters for the model such that they maximize the log likelihood calculated in the expectation step.

[12] `http://bnt.googlecode.com/`

The process stops once it converges on a log likelihood for the data given the model's parameters.

10.2.2 Subselecting the Number of Observations

As we increase the number of observation random variables, so do we increase the number of conditional probability tables whose values we need to learn using the expectation maximisation process described in Section 10.2.1. As this is an expensive procedure, we would like to use the minimal number of observations needed to make good predictions. One way of selecting which observations to use is to apply a forward selection algorithm. The application of such an algorithm selects a subset of observations $\{O_i, , ..., O_p\}$ from the set of all observations $\{O_1, O_2, ..., O_n\}$ as follows. We call this subset of observations the *selected observations*. The algorithm's steps are as follows:

1. Iterate through the possible observation variables. For each observation variable, build a latent state-space model using just the state random variables and that observation random variable.

2. Evaluate the performance of each model and determine which one yielded the best performance. The observation random variable that was used to build this model is selected as the first observation in the set of selected observations.

3. Iterate through the remaining observations. For each remaining observation random variable, build a latent state-space model using it and the previously selected observation set.

4. Evaluate the performance of these new models and select the best one. The random observation variables it uses become the new selected observation set.

5. Repeat steps three and four, building models with the random observation variables in the selected observations set in addition to one random observation variable not yet selected. Continue to grow the selected observation set like this until performance does not improve.

10.2.3 Selecting the Number of Hidden States

We cannot directly observe the state a patient is in. For this reason we do not know what the number of possible states can be. Yet to build the latent state-space models we need to select the number of possible hidden states, that is, we need to specify the support of the state random variables. To determine the support of the state random variables we choose a range for the possible supports and build models using support values from that range. We then select the support based on which model was the most likely fit as determined by the greatest log likelihood.

10.3 Transforming Data Into Learning Samples

As mentioned in Section 9.2, before we can build a model we must first transform our data into learning samples. We transform the blood pressure problem into learning samples in three steps:

1. First, we model a beat-sample time series as a series of state transitions. A discrete state corresponds to a time slice of the beat-sample time series, and the state stays constant during this time slice.

2. Second, we apply the aggregate functions defined in Section 8.3.1 to each time slice of the beat-sample time series. The aggregate features yielded by the aggregate functions are the observations of the state corresponding to that particular time slice of the beat-sample time series.

3. Third, we discretize the observations and determine the cardinalities of the random variables $V_{i,t}$.

To demonstrate how one creates such a latent state-space model using these steps, and to introduce how we discretize the observations, we give a simple example. We model the example b_{map} time series given in Table 1 in Section 9.1 using a state duration of two seconds.

Step one of the modelling process is to model the beat-sample time series as state transitions. As the example time series is 10 seconds long, we require 4 state transitions to model it. The grouping of b_{map} time-samples by state duration is as follows, where each column indicates the time duration during which the state stays constant, the separators between the columns indicate when state transitions occur and the cell contents gives the sample values:

States	H_1	H_2	H_3	H_4	H_5
b_{map} time series	70,65	80,74	68,65,50	60,63	70,80

Step two of the modelling process is to apply the aggregate functions to each state's time slice of the beat-sample time series. To keep the example simple, we only demonstrate the application of the aggregate-mean and aggregate-standard-deviation functions. The application of the aggregate functions to each state duration of b_{map} time series yields the following observations:

States	H_1	H_2	H_3	H_4	H_5
m_t Observation	67.5	77	61	61.5	75
sd_t Observation	3.54	4.24	9.64	2.12	7.07

Step three is to discretize the observations. This is done by choosing a number of bins and then seeing into which range the observations fall. There are different ways to choose the boundary of these bins; we decided on the following method:

- For the aggregate-mean observation we use the same boundaries as the blood pressure class bins defined in Section 4. Since there are ten different blood pressure classes, this observation will be represented by 10 values.

- For the other observations we use 10 bins as well. To determine their boundaries we use the following method:

  - From the time series belonging to the learning sample, we extract all observations for the six observation types other than mean. This gives us six different sets of extracted observations: SD, SK, KU, TR, V, A.

- For each of these sets we calculate the mean μ and standard deviation σ. We thus have SD_μ, SK_μ, KU_μ, TR_μ, V_μ, A_μ and SD_σ, SK_σ, KU_σ, TR_σ, V_σ, A_σ
- For each observation type we set the boundaries of its bin as follows:
    - The left boundary of the leftmost bin is set to $-inf$
    - The right boundary of the leftmost bin is set to $\mu - 2\sigma$
    - The left boundary of the rightmost bin is set to $\mu + 2\sigma$
    - The right boundary of the rightmost bin is set to inf
    - The boundaries of all the other bins are spaced evenly between $\mu - 2\sigma$ and $\mu + 2\sigma$

The consequence of this step is that each type of observation will be represented as a discrete random variable with a cardinality equal to the number of bins used for discretization. Furthermore, each learning sample will have its own set of bins for each non aggregate-mean feature. We name these random variables after the observations they represent: V_m, V_{sd}, V_{sk}, V_{ku}, V_{tr}, V_v, and V_a. As we use the same number of bins for discretization of each type of observation, the cardinality for all V_i random variables is 10.

To continue our example from above, we can calculate the bin boundaries to discretize V_{sd}. As SD_μ and SD_σ are 5.32 and 3.01, respectively, the discretization boundaries of the ten bins are:

1	-inf	-0.7030
2	-0.7030	0.8032
3	0.8032	2.3095
4	2.3095	3.8158
5	3.8158	5.3220
6	5.3220	6.8282
7	6.8282	8.3345
8	8.3345	9.8407
9	9.8407	11.3470
10	11.3470	inf

Using these boundaries, we determine the observations to be:

States	H_1	H_2	H_3	H_4	H_5
m_t Observation	4	6	3	3	5
sd_t Observation	4	5	8	3	7

10.4 Latent State-Space Model Inference

Once we've learned the conditional probabilities of all CPTs in the latent state-space model we can use the model to make inferences. We demonstrate how the model makes an inference of a future observation by means of an example. For this example we use the latent state-space model depicted in Figure 12, which consists of 4 time slices. Assume we are given a beat-feature time series for the first two time slices. We wish to infer what the blood pressure will be in the fourth time slice. We furthermore assume that we have determined the support of the state random variables to be 5 and selected the observation

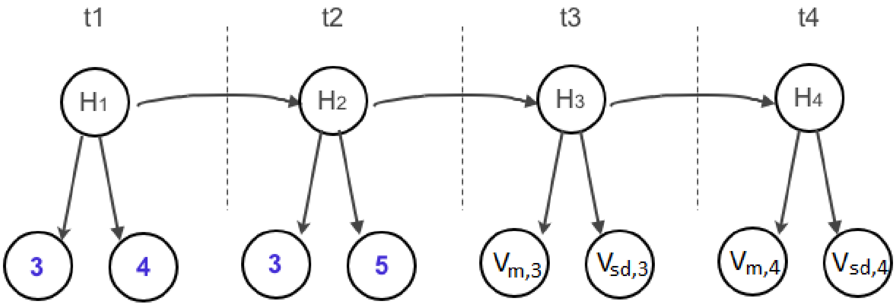


Figure 13: Latent state space model with evidence provided

random variables to be the aggregate-mean ($V_{m,t}$) and the aggregate-standard-deviation ($V_{sd,t}$). Inference of the blood pressure is performed in three steps. These steps are depicted in Figures 13, 14, 15 and are as follows:

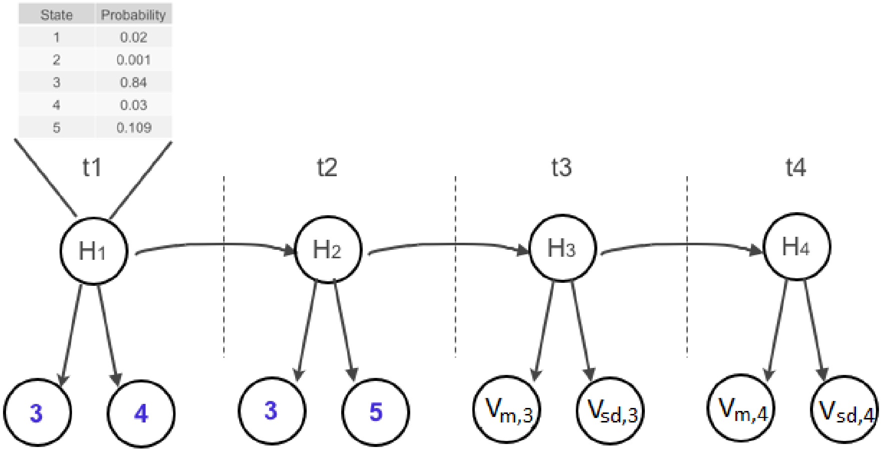
1. In a first step, the observations are extracted from the beat-feature time series. As the beat-feature time series is of duration two time slices, we can provide the model with the values taken by $V_{m,1} = 3$ $V_{sd,1} = 4$ $V_{m,2} = 3$ $V_{sd,2} = 5$.

2. Using the provided evidence and bayes formula, the model calculates the conditional probability distributions of the random variables that represent the state of the patient. For example, in time slice t1, the first table entry gives the probability 0.02 that the state random variable H_1 takes value 1 given the observations.

3. The calculated conditional probability distribution of H_2 can be used as indicators when calculating the maximum likelihood of $V_{m,4}$, the aggregate-mean blood pressure of the patient in time slice 4. The details for how one infers a future state given indicators is not discussed in this thesis, but can be found in [25].

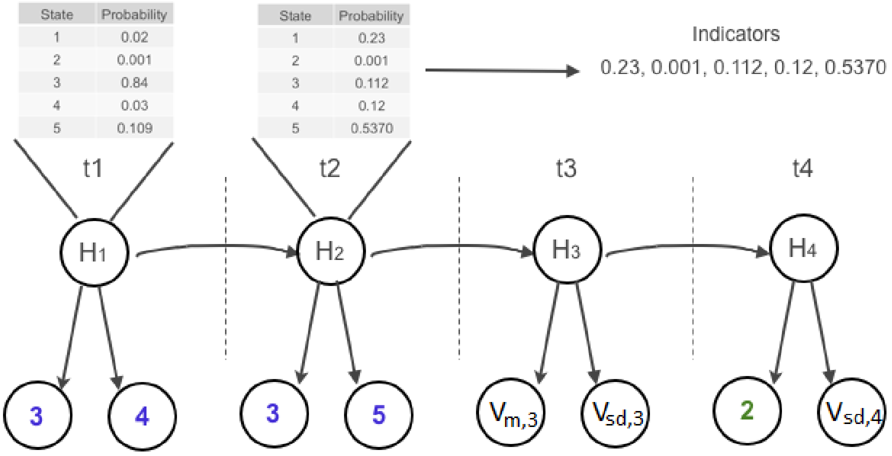
10.5 Advantages of Latent State-Space Models

Latent state-space models provide several advantages over static models. First, they allow for flexible predictions; once the the conditional probability tables have been learned, evidence can be provided for any number of time slices, and inferences as to the maximum likelihood of any observation random variable in any future time slice can be made. This allows the model to make predictions for varying lead times. Second, the states in a latent state-space model represent unobservable conditions. Although it is unknown to us which unobservable conditions they represent, it is still possible to use the latent state-space model to infer the most likely state that a state random variable will take. This probabilistic information may allow us to draw conclusions as to what the states might represent. In future work it might be possible to cross reference these probability distributions with clinical data in the MIMIC-II clinical data set.



State	Probability
1	0.02
2	0.001
3	0.84
4	0.03
5	0.109

Figure 14: Latent state space model with probability distribution of H_1 calculated



State	Probability
1	0.02
2	0.001
3	0.84
4	0.03
5	0.109

State	Probability
1	0.23
2	0.001
3	0.112
4	0.12
5	0.5370

Indicators

0.23, 0.001, 0.112, 0.12, 0.5370

Figure 15: Latent state space model with probability distribution of H_1, H_2 calculated

11 Experiments

Our overall goal is to be able to predict blood pressure in the most general case. The way to show how well our models are able to predict in the most general case would be to evaluate them on the entire MIMIC-II ABP dataset. However, for the reasons discussed in Section 8.2, we choose to train and evaluate our data on the 1000-record-subset.

During evaluation we will will frequently make use of the so called f1 score as well as the class deviation. In Section 11.1 we briefly summarize these metrics as well as make an argument as to why it makes sense to employ them to evaluate our experiment.

As there are inherent differences between the two different model types, the blood pressure prediction experiment cannot be set up in the same way for both. We discuss how we set up the blood pressure prediction problem for static models and latent state-space models in Section 11.2. One of our goals with respect to performance is to compare the two different model types. How one can compare them despite the difference in their experimental setup is discussed in Section 11.3. To understand how well the models perform, it is not enough to evaluate performance metrics or compare the two models against each other; it is necessary to compare their performance against a baseline. Our baseline for the blood pressure experiment is discussed in Section 11.4.

11.1 Evaluation Metrics

We use two different error metrics during evaluation. The first is the f1 score, the second is an evaluation metric of our own definition, the average class deviation.

11.1.1 f1 Score

When the class distribution is uneven, as is the case in the blood pressure experiment, the performance of a classifier should not be measured in terms of accuracy. This is because a classifier that is biased towards predicting a class that appears frequently in the class distribution will yield a high accuracy. This high accuracy shrouds the classifiers performance on the other, less frequent classes. For this reason, a better representation of a classifier's performance in the case of an uneven class distribution is given by the precision and recall performance metrics, which are defined for each class c_i as:

$$Precision_{c_i} = \frac{\text{\# of class } i \text{ identified correctly}}{\text{total declared as class } i} \tag{3}$$

$$Recall_{c_i} = \frac{\text{\# of class } i \text{ identified correctly}}{\text{total number of examples of class } i} \tag{4}$$

As measuring two different performance metrics makes for unwieldy comparison, we decided to use the f1 score metric. The f1 score is a combination of precision and recall, and is defined as:

$$f1Score_{c_i} = \frac{2 * Precision_{c_i} * Recall_{c_i}}{Precision_{c_i} + Recall_{c_i}} \tag{5}$$

One issue when employing the f1 score is that if a certain class is never declared by the classifier, then the precision, and therefore the f1 score, is undefined. We make the compromise that, if the f1 score is undefined, we declare it to be 0 during our evaluation. We justify this decision in that our goal is to evaluate the overall classifier performance on each class. We decide that if a classifier never declares a certain class, we evaluate its performance on that class as poor (0).

11.1.2 Average Class Deviation

It is not always convenient to use the f1 score as an evaluation metric, and, in particular, not for evaluating the prediction accuracy of the latent-state-space-model changes over consecutive time slices. In this case it would be necessary to track ten different f1 scores over several time slices. Tracking so many numbers makes it difficult to interpret the results. For this reason we define a metric called the *average class deviation.* We define it as:

$$\textit{avg-class-dev} = \frac{1}{n} * \sum_{i=1}^{n} |c_{i,t} - c_{i,p}| \tag{6}$$

where n is the total number of test samples, $c_{i,t}$ is the target, i.e., ground-truth, class label and $c_{i,p}$ is the predicted class label. As this metric represents the average distance from the correct class, a smaller average class deviation indicates a better classifier performance.

11.2 Experimental Setup

11.2.1 Static Model Experiment Setup

We set the static model experiment up in the following manner. We define the goal to be to predict the average blood pressure of a patient over a period of 20 minutes (prediction window) starting at 60 minutes in the future from the current time (lead), using 20 minutes of history (lag). To create a training and test set we extracted the maximum number of possible samples from the 1000 patient database using the procedure described in Section 9.1. From this data set we selected 70% at random to make up our training set, regardless of which patient the samples belonged to. The remaining 30% were set aside as the test set. An overview of the experimental parameters is given in the following table.

γ	20 min
τ	60 min
α	20 min
Total # of learning samples	95,256
Total # of training samples	66,679 (70%)
Total # of testing samples	28,577 (30%)

We selected the range of parameters over which our Multi-parameter, Multi-algorithm Suite would search in such a way that the complete experiment could run in a couple of days when distributed to 80 nodes on the cloud. The number of the classifier instances per algorithm that were created and trained is given in the following table. Each classifier instance uses a different set of parameters.

Classifier type	# of classifier instances
Neural Networks	720
Discrete Bayesian Networks	18
Naive Bayes	64
Support Vector Machines	100
Decision Trees	324

From running this experiment we would like to be able to draw conclusions about:

1. The possible performance of any classification algorithm on our problem definition using the features we extracted. We will be able to do this because we cover all feasible configurations of all standard machine learning algorithms.

2. The parameters and the performance of the 'best' classifier. We will evaluate all the classifiers on the training set using 10-fold cross validation. We will select the classifier whose f1 score on class 1 is the greatest when the classifier is evaluated on the training set using cross validation. We chose class 1 because predicting low blood pressure would be useful information for ICU staff.

11.2.2 Latent State-Space Model Experiment Setup

We set the latent state-space model experiment up as follows. We build a 12 time slice model of state and observation random variables. The set of state random variables is therefore $\{H_1, H_2, ..., H_{12}\}$, and the set of observation random variables is $\{V_{m,1}, V_{sd,1}, ..., V_{a,1}, ..., V_{m,12}, V_{sd,12}, ..., V_{a,12}\}$. We set the duration of each time slice to be 20 minutes, so the entire state space model we build models for has a total duration of four hours. Whenever we evaluate a model's performance we provide evidence for the observations of the first six time slices. We then have the model infer the aggregate-mean observations for time slices 7-12 and compare them to the actual aggregate-mean observations.

Because we are building a model for four hours, we extract all contiguous four hour segments from the 1000 patients. The total number of segments extracted was 39,005. Each segment overlaps at most 11 out of the 12 time slices of any other segment. From this data set of four hour segments, we select without replacement 1000 at random for training and 1000 at random for testing. The reason the training data set is so much smaller here than in the static case is that training a latent state-space model is computationally expensive. Even when training was parallelized and run on a single system having eight cores, model computation took half a day. The reason for not selecting all the remaining segments for testing was that we wanted only a small number of segments to overlap.

We select the parameters, selected observations set and support of the state random variables using feature forward selection and log likelihood fit in that order, as described in Section 10.2. As we intended to make inferences on the aggregate-mean observation random variables, we forced the feature selection algorithm to choose the aggregate-mean random variable first, regardless of it's performance. Evaluation of models in the feature forward selection is performed by three-fold cross validation on the training set. At this point we arbitrarily set the initial value of the support for the state random variables to 18.

The search for the support of the state random variables covers the range [2..20]. Log Likelihood fit is calculated as the average log likelihood when performing three-fold cross validation on the training data.

The table below provides an overview of the experimental setup.

Number of time slices in model	20 minutes
Duration of time slice	20 minutes
Total number of extracted 4 hour segments	95,256
Size of training set	1000 four hour segments
Size of testing set	1000 four hour segments
Number of inferences per testing time segment	6

11.3 Performance Comparison

Although there are differences in the way we set up experiments for the two types of models, it is still possible to draw performance comparisons between the two. Our goal is to compare the models by f1 score performance across all 10 classes. We do this by comparing the f1 score performance of the 'best' classifier, chosen by the criteria defined in Section 11.2.1, to the f1 score performance of the latent state-space model when predicting the observation of time slice 10, which corresponds to making a prediction for the prediction window 60-80 minutes in the future.

11.4 A Baseline to Compare Against

When evaluating predictive models it is important to compare them against a baseline. The absolute values of a model's performance metrics are not very meaningful without some reference to compare them against. It might be that a similar performance can be achieved using a trivial strategy. Such a trivial strategy provides a baseline against which we can compare our models. One such trivial strategy is a model that makes a prediction at random. If the distribution of class labels is uneven, as is the case in our blood pressure prediction experiments, this random strategy can be adapted such that the classifier makes predictions at random using the a-priori class distribution learned from the training set. It is common practice to use a random strategy as a baseline.

In the case of blood pressure prediction, however, there is a trivial, intuitive strategy that performs well. This is the persistent strategy. The idea behind this strategy is, given a person's blood pressure, there is a good reason to believe his blood pressure will not change in the near future. Using this idea we define a 20 minute lag persistence classifier to be a classifier that, when presented with a b_{map} beat-feature signal, extracts the aggregate mean from it, determines which class the extracted mean corresponds to and predicts the class label will not change in the future. When the 20 minute lag persistence classifier is used as a baseline against which static models are compared, the 20 minutes belonging to the lag period are used as the input for the 20 minute lag persistence classifier. For the latent state-space model, the 20 minutes of signal belonging to the 6th time slice are used by the 20 minute lag persistence classifier. For example, if the blood pressure in the sixth time slice corresponds to class three, than the

20 minute lag persistence classifier will predict class three for time slices seven through twelve.

12 Results

12.1 Static Models

In Figure 16 we show the distribution of f1 scores across all static models built by our Multi-Algorithm, Multi-Parameter Suite. From the set of all static models trained we select the 'best' one. As described in Section 11.2.1, our selection for 'best' is the greatest f1 score on class label 1 when performing cross validation on the training data. Our results yielded that the 'best' static model was built using a neural network with a hidden layer size of 40. A radial-basis transfer function was selected as the transfer function for the neurons in the hidden layer. In Figure 17 we show the distribution of f1 scores across all classes when the 'best' static model is evaluated on the test set. As a comparison we also plot the baseline, that is, the results of the 20 minute lag persistence classifier.

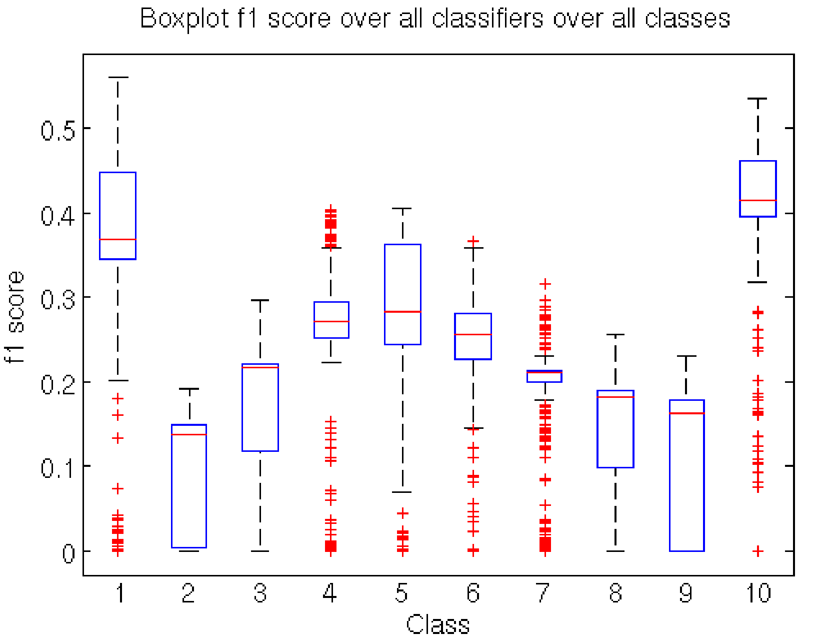


Figure 16: Distribution of f1 score across all classes and all classes

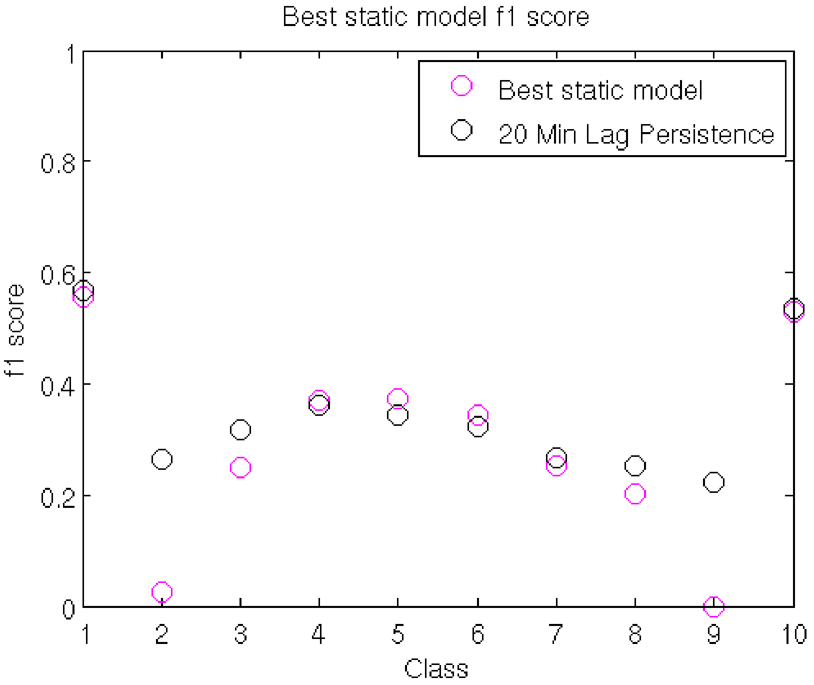


Figure 17: f1 scores across all classes for the 'best' static model

12.2 Latent State-Space Models

12.2.1 Training

In the training phase, the forward feature selection algorithm selected observations corresponding to the aggregate skew (sk_t) in addition to the aggregate mean (m_t). Adding additional observations did not improve the performance of the latent state-space model.

After selecting features, the support of the state random variables was varied and, for each value, latent state-space models were created and the log-likelihood of the training data fitting the model was calculated using three-fold cross validation. The measurements of the log-likelihood fit scores are plotted in Figure 18. It turned out that a model with a state random-variable support of 19 yielded the best fit.

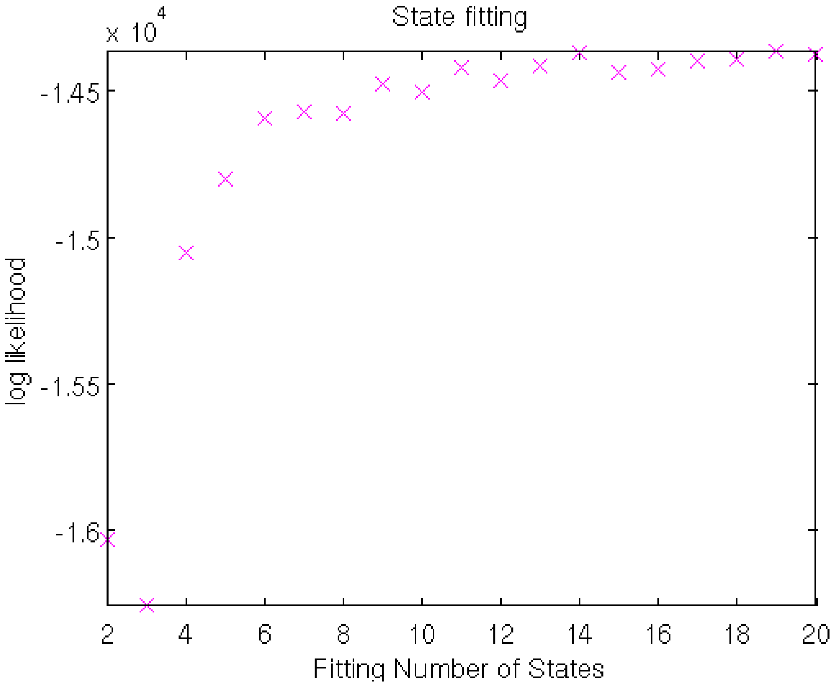


Figure 18: Log-likelihood fit of models with varying number of states

12.2.2 Testing

After training a latent space model we used it to make inferences on the 10th time slice. This corresponds to predicting 60 minutes into the future. We plot the performance on the different class labels of the test set as the f1 score in Figure 19. As a baseline, we plot the f1 score performance of the 20 min lag persistence classifier.

Finally, we use the trained latent state-space model to infer the blood pressure during time slices 7 through 12. We plot the avg. class deviation for each of these inferences along with the average class deviation (*avg-class-dev*) of the 20 minute persistence classifier in Figure 20.

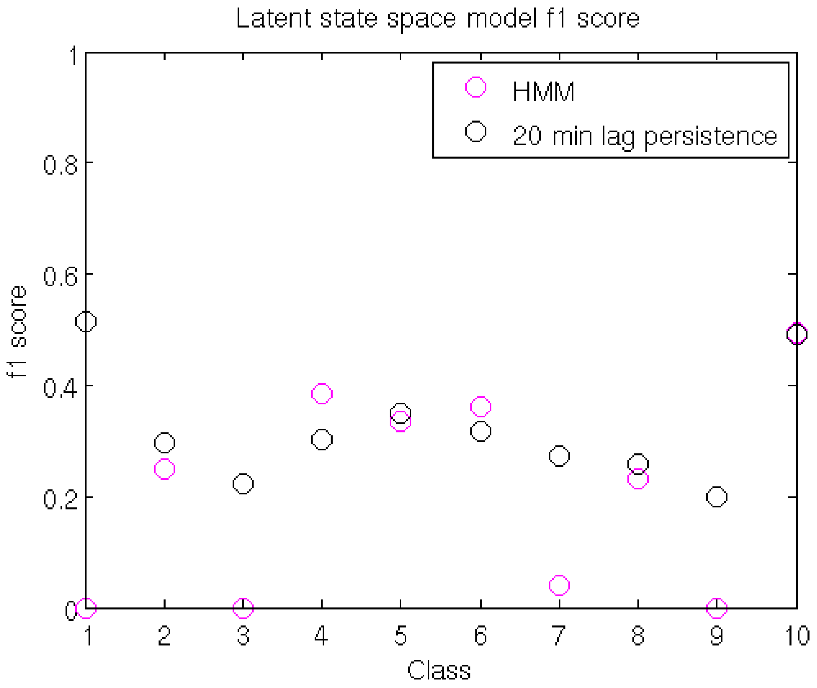


Figure 19: f1 scores across all classes for the latent state space model

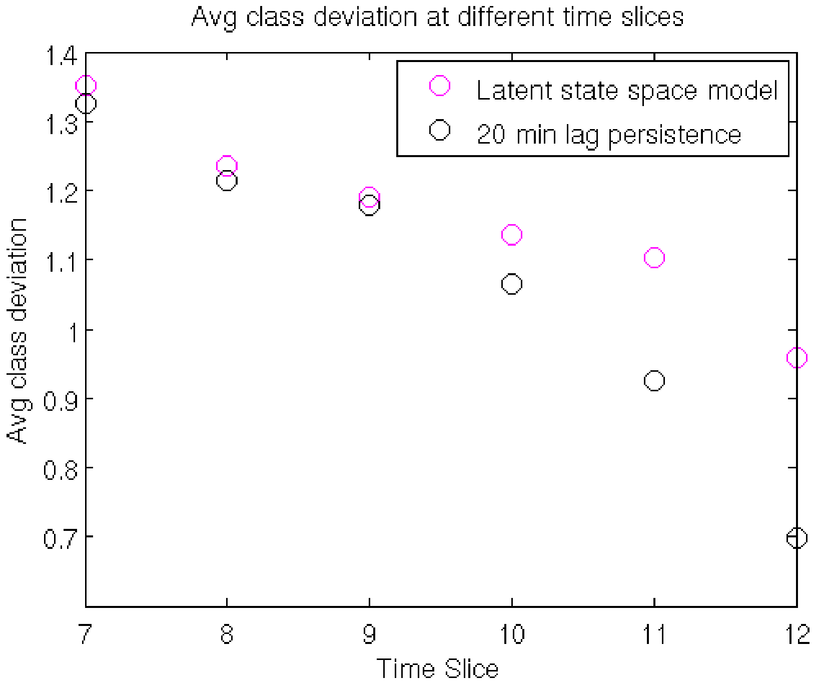


Figure 20: Avg class deviation at different time slices.

13 Discussion and Conclusions

From the plots in Section 12 we see that both types of models we built never outperform the 20 minute lag persistence classifier. The consequence of these results means that many questions we posed in the first half of Section 11 remain unanswered. Furthermore, it is difficult to compare our static models to our latent state-space models as both perform poorly when predicting a 20 minute prediction window 60 minutes in the future. We built and evaluated our models with care:

- By flagging abnormal beats with the signal abnormality index we make sure no noisy data is used when extracting both features and class labels from arterial blood pressure data.
- By ensuring all training and testing data segments are contiguous segments without any gaps.
- By using a large amount of data for training and for evaluation we ensure that our models are exposed to many different types of patients during training and testing.
- By choosing a baseline for our experiments and evaluating our models against it using metrics that make sense when the class distribution is uneven, we are able evaluate how good our models actually are and if they perform better than a simple persistance strategy.
- By designing and applying our Multi-Algorithm, Multi-Parameter Suite to model the problem with a whole set of classifiers covering all possible parameters, we are certain that we have not missed a parameter configuration that would drastically improve performance.

The end result of this thesis is a set of carefully built systems that will be used in future work. In particular, we will use them to explore whether or not predicting blood pressure using simple signal features is difficult. The results in this work seem to indicate that it is indeed difficult, but before we can conclude this with certainty, several improvements will be made to the systems (see also Section 14):

- We will add additional simple beat features.
- We will enhance the latent state-space model with the following features:
  - We will use continuous instead of discrete random variables for the observation random variables. We will experiment with different distributions for these random variables, including Gaussian and non-Gaussian distributions.
  - We will modify the learning structure so that it fits the data better.
- Finally, we will perform more thorough testing with patients from the entire MIMIC-II ABP waveform dataset.

14 Future Work

The results in Section 12 show that there is still work to be done to achieve our goal of being able to predict blood pressure. Moreover, the work we have done so far indicates the direction in which future work should proceed. In particular, we see four different areas that should be explored further: exploring and grouping data, improving algorithms, adding features and refining data extraction.

To better understand why our approach did not yield the kind of results we desire, it is necessary to get a better understanding of the kind of data we trained and tested our models on. That is, it is necessary to *explore and group the data*. Our goal is to build a model that can predict blood pressure for the general case. For this reason, we used data from a large subset of the data in the MIMIC-II database. This data contains a large variety of different kinds of patients. For example, some of the patients with low blood pressure may be patients who are in a deep coma and whose blood pressure remains low constantly. Others may belong to patients whose blood pressure fluctuates due to an infection or the administration of medication. Our models were trained using data from both kinds of patients, and one reason for their poor performance may be that they are trying to predict a middle ground between patient groups. If we train and evaluate our performance on specific subsets of the data, we may find that they yield a greater performance. If this is the case, it would also help to identify which subsets of patients have a blood pressure development that is particularly hard to predict. Possible subsets of patients are: patients whose blood pressure is very volatile or very stable, patient's who have a 'normal' blood pressure that is low, patient's that have suffered from a previous hypotensive event and patient's that currently have an infection.

The step of *exploring and grouping data* leads into the next step, namely, *improving algorithms*. If we find that the performance of our models improves on specific subsets of patients, we could improve the performance of our models on general data by building a model that predicts blood pressure in two steps. In a first step, classifiers that have been trained to identify which subset of patients a generic patient belongs to provide a probability estimate as to which subset of patients a new patient belongs to. In a second step, blood pressure prediction models that are specialized for each subset of patients make a probabilistic estimate as to what the likelihood of each blood pressure class is. The blood pressure class that is then predicted is a function of both sets of probabilities. In addition to building such two step algorithms, we should explore the performance of machine learning algorithms that predict blood pressure as a continuous value.

Finally, we should continue to improve our beat database in the form of adding additional beat features and refining beat extraction. For beat features there are many different kinds one can imagine extracting from a beat. Some of these features are in the time domain, such as skew, kurtosis and the dynamic time warping distance measure to the previous beat or to an 'average' beat. Other features are in the time-frequency domain, such as fast Fourier transforms or wavelet transforms. As to *refining beat extraction*, the second step of the beat feature database construction pipeline described in Section 6.2 has not yet been implemented. The purpose of this step is to clean the ABP waveform before extracting beat onsets and beat features from the waveform. Implementation of this step will allow us to extract cleaner beats, something that is beneficial when extracting more sophisticated beat features. The challenge in implementing this step stems from the fact that simple filters, such as low pass filters, may remove high frequency information that could be captured by specific beat features. For this reason, we aim to explore different types of filter techniques in order to find one with the right tradeoff between removing noise and retaining high frequency information from ABP waveforms.

15 References

[1] The mgh/mf waveform database. `http://physionet.org/pn3/mghdb/`. Accessed: 2013-06.

[2] Mimic ii databases. `http://physionet.org/mimic2/`. Accessed: 2013-05.

[3] Mimic ii: Record matching and surrogate dates. `http://physionet.org/mimic2/mimic2_matching.shtml`. Accessed: 2013-05.

[4] The mimic ii waveform database matched subset. `http://physionet.org/physiobank/database/mimic2wdb/matched/`. Accessed: 2013-05.

[5] X Chen, D Xu, G Zhang, and R Mukkamala. Forecasting acute hypotensive episodes in intensive care patients based on a peripheral arterial blood pressure waveform. In *Computers in Cardiology, 2009*, pages 545–548. IEEE, 2009.

[6] F Chiarugi, I Karatzanis, V Sakkalis, I Tsamardinos, Th Dermitzaki, M Foukarakis, and G Vrouchos. Predicting the occurrence of acute hypotensive episodes: The physionet challenge. In *Computers in Cardiology, 2009*, pages 621–624. IEEE, 2009.

[7] Aram V Chobanian, George L Bakris, Henry R Black, William C Cushman, Lee A Green, Joseph L Izzo, Daniel W Jones, Barry J Materson, Suzanne Oparil, Jackson T Wright, et al. Seventh report of the joint national committee on prevention, detection, evaluation, and treatment of high blood pressure. *Hypertension*, 42(6):1206–1252, 2003.

[8] Colleen M Ennett, KP Lee, Larry J Eshelman, Brian Gross, Larry Nielsen, Joseph J Frassica, and Mohammed Saeed. Predicting respiratory instability in the icu. In *Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE*, pages 2848–2851. IEEE, 2008.

[9] PA Fournier and JF Roy. Acute hypotension episode prediction using information divergence for feature selection, and non-parametric methods for classification. In *Computers in Cardiology, 2009*, pages 625–628. IEEE, 2009.

[10] John Gade, Ilkka Korhonen, Mark van Gils, Peter Weller, and Leena Pesu. Technical description of the ibis data library. *Computer methods and programs in biomedicine*, 63(3):175–186, 2000.

[11] Marzyeh Ghassemi. Methods and models for acute hypotensive episode prediction. Master's thesis, University of Oxford, 2011.

[12] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[13] David A Harrison, Gareth J Parry, James R Carpenter, Alasdair Short, and Kathy Rowan. A new risk prediction model for critical care: The intensive care national audit & research centre (icnarc) model*. *Critical care medicine*, 35(4):1091–1098, 2007.

[14] JH Henriques and TR Rocha. Prediction of acute hypotensive episodes using neural network multi-models. In *Computers in Cardiology, 2009*, pages 549–552. IEEE, 2009.

[15] Thomas L Higgins, Daniel Teres, and Brian Nathanson. Outcome prediction in critical care: the mortality probability models. *Current opinion in critical care*, 14(5):498–505, 2008.

[16] SM Jakob, K Nieminen, J Hiltunen, J Karhu, and J Takala. Ibis data library: Clinical description of the finnish database. *Computer methods and programs in biomedicine*, 63(3):161–166, 2000.

[17] F Jousset, M Lemay, and JM Vesin. Computers in cardiology/physionet challenge 2009: Predicting acute hypotensive episodes. In *Computers in Cardiology, 2009*, pages 637–640. IEEE, 2009.

[18] Ilkka Korhonen, Jyrki Ojaniemi, K Niieminen, Mark Van Gils, Arno Heikela, and Aarno Kari. Building the improve data library. *Engineering in Medicine and Biology Magazine, IEEE*, 16(6):25–32, 1997.

[19] P Langley, ST King, D Zheng, EJ Bowers, K Wang, J Allen, and A Murray. Predicting acute hypotensive episodes from mean arterial pressure. In *Computers in Cardiology, 2009*, pages 553–556. IEEE, 2009.

[20] Qiao Li, Roger G Mark, and Gari D Clifford. Robust heart rate estimation from multiple asynchronous noisy sources using signal quality indices and a kalman filter. *Physiological measurement*, 29(1):15, 2008.

[21] Qiao Li, Roger G Mark, Gari D Clifford, et al. Artificial arterial blood pressure artifact models and an evaluation of a robust blood pressure and heart rate estimator. *Biomedical engineering online*, 8(1):13, 2009.

[22] GF Mandersloot, RC Pottinger, PR Weller, PF Prior, C Morgan, NJ Smith, and RM Langford. The ibis project: data collection in london. *Computer methods and programs in biomedicine*, 63(3):167–174, 2000.

[23] MA Mneimneh and RJ Povinelli. A rule-based approach for the prediction of acute hypotensive episodes. In *Computers in Cardiology, 2009*, pages 557–560. IEEE, 2009.

[24] GB Moody and LH Lehman. Predicting acute hypotensive episodes: The 10th annual physionet/computers in cardiology challenge. In *Computers in Cardiology, 2009*, pages 541–544. IEEE, 2009.

[25] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*. The MIT Press, 8 2012.

[26] K Nieminen, Richard M Langford, CJ Morgan, J Takala, and Aarno Kari. A clinical description of the improve data library. *Engineering in Medicine and Biology Magazine, IEEE*, 16(6):21–24, 1997.

[27] PR Norris and BM Dawant. Closing the loop in icu decision support: physiologic event detection, alerts, and documentation. In *Proceedings of the AMIA Symposium*, page 498. American Medical Informatics Association, 2001.

[28] A Pérez-Silva and JL Merino. Frequent ventricular extrasystoles: significance, prognosis and treatment.

[29] Jan Ramon, Daan Fierens, Fabián Güiza, Geert Meyfroidt, Hendrik Blockeel, Maurice Bruynooghe, and Greet Van Den Berghe. Mining data from intensive care patients. *Advanced Engineering Informatics*, 21(3):243–256, 2007.

[30] Marta L Render, James Deddens, Ron Freyberg, Peter Almenoff, Alfred F Connors Jr, Douglas Wagner, and Timothy P Hofer. Veterans affairs intensive care unit risk adjustment model: Validation, updating, recalibration*. *Critical care medicine*, 36(4):1031–1042, 2008.

[31] Mohammed Saeed, C Lieu, G Raber, and RG Mark. Mimic ii: a massive temporal icu patient database to support research in intelligent patient monitoring. In *Computers in Cardiology, 2002*, pages 641–644. IEEE, 2002.

[32] Mohammed Saeed, Mauricio Villarroel, Andrew T Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H Kyaw, Benjamin Moody, and Roger G Mark. Multiparameter intelligent monitoring in intensive care ii (mimic-ii): a public-access intensive care unit database. *Critical care medicine*, 39(5):952, 2011.

[33] Suchi Saria, Daphne Koller, and Anna Penn. Learning individual and population level traits from clinical temporal data. In *Proc. Neural Information Processing Systems (NIPS), Predictive Models in Personalized Medicine workshop*, 2010.

[34] Suchi Saria, Anand K Rajani, Jeffrey Gould, Daphne L Koller, and Anna A Penn. Integration of early physiological responses predicts later illness severity in preterm infants. *Science translational medicine*, 2(48):48ra65, 2010.

[35] JX Sun, AT Reisner, and RG Mark. A signal abnormality index for arterial blood pressure waveforms. In *Computers in Cardiology, 2006*, pages 13–16. IEEE, 2006.

[36] Zeeshan Syed, John Guttag, and Collin Stultz. Clustering and symbolic analysis of cardiovascular signals: discovery and visualization of medically relevant patterns in long-term data using limited prior knowledge. *EURASIP Journal on Applied Signal Processing*, 2007(1):97–97, 2007.

[37] Zeeshan Syed, Collin Stultz, Manolis Kellis, Piotr Indyk, and John Guttag. Motif discovery in physiological datasets: a methodology for inferring predictive elements. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 4(1):2, 2010.

[38] Carsten E Thomsen, Luc Cluitmans, and Tarmo Lipping. Exploring the ibis data library contents: tools for data visualisation,(pre-) processing and screening. *Computer methods and programs in biomedicine*, 63(3):187–201, 2000.

[39] Carsten E Thomsen, John Gade, Kari Nieminen, Richard M Langford, I Robert Ghosh, Kjeld Jensen, M Van Gils, Annelise Rosenfalck, Pamela Prior, and Steven White. Collecting eeg signals in the improve data library. *Engineering in Medicine and Biology Magazine, IEEE*, 16(6):33–40, 1997.

[40] Peter Weller, Tom Hennessy, and Ewart Carson. The ibis system architecture. *Computer methods and programs in biomedicine*, 63(3):229–235, 2000.

[41] Jack E Zimmerman and Andrew A Kramer. Outcome prediction in critical care: the acute physiology and chronic health evaluation models. *Current opinion in critical care*, 14(5):491–497, 2008.

[42] W Zong, T Heldt, GB Moody, and RG Mark. An open-source algorithm to detect onset of arterial blood pressure pulses. In *Computers in Cardiology, 2003*, pages 259–262. IEEE, 2003.

[43] W Zong, GB Moody, and RG Mark. Reduction of false arterial blood pressure alarms using signal quality assessement and relationships between the electrocardiogram and arterial blood pressure. *Medical and Biological Engineering and Computing*, 42(5):698–706, 2004.