beatDB: A Large Scale Waveform Feature Repository

Franck Dernoncourt, Kalyan Veeramachaneni, Una-May O'Reilly
CSAIL, MIT
{francky, kalyan,unamay}@csail.mit.edu;

1 Introduction

For typical physiological waveform studies, researchers define a study group within which they designate case and controls. They extract the group's waveforms, filter the signals, pre process them and extract features before iteratively executing, evaluating and interpreting a pre-selected machine learning algorithm with metrics such as area under the curve and analyses such as variable sensitivity.

Recognizing that a typical study, even with modest quantities of patients, can take 6 to 12 months, we have asked how this duration can be shrunk and multiple studies can share development effort. In response we are designing a large scale machine learning and analytics framework, named *beatDB* , for mining knowledge from high resolution physiological waveforms. BeatDB is our first cut at creating very large, open access, repositories of feature-level data derived from continuous periodic physiological signal waveforms such as electro-cardiograms (ECG) or arterial blood pressure. We have presently processed close to a billion arterial blood pressure beats from MIMIC 2 version 3 waveform database and developed a strategy for feature extraction and discovery which supports efficient data studies. Thus, beatDB radically shrinks the time of large scale investigations by judiciously pre-computing beat features which are likely to be frequently used. It supports agile investigation by offering parameterizations that allow task specific compute and storage tradeoff decisions to be made when computing additional features and preparing data for machine learning or visualization. It resolves questions such as: what data abstractions of waveforms are best for efficient data storage, retrieval and further processing? What features should be pre-processed and stored vs computed "on the fly"? What interfaces to machine learning and analytics are generally useful? Where should the process be agile and modular? How can large scale data be processed speedily and efficiently? In this paper, we briefly describe beatDB and demonstrate how it supports hypothesis testing on the entire set of arterial blood pressure data in the MIMIC 2 database.

2 BeatDB

BeatDB addresses a number of potential bottlenecks which occur with large scale physiological data, i.e. the pre-processing step where signal noise is identified and/or repaired, feature extraction, and creating aggregates while dealing with gaps in a time series segment. We first identified **efficient data abstractions**. Physiological waveforms are periodic with samples which form *beats*. Prior to modeling, most waveform analyses transform *raw* waveforms $x(t)$ into a time series of *features* $f(t)$ extracted on a per beat basis to covariates $V_{1...m}$ which are derived via a variety of aggregation methods [1]. While there is a lot of consensus in terms of what *features* are extracted from each beat, there is a lot variety in the aggregation of feature times series into covariates. This arises because different machine learning algorithms expect data in different representations. For example, dynamic bayesian networks expect an aggregated time series while a decision tree expects covariates that can be independent of time.

To facilitate efficient pre-processing we had to deal with frequent breaks in the data which arise because of patient movement and sensor sensitivity. These breaks make a series falsely contiguous and require it to be separated into multiple parts beatDB calls *segments*. Additionally, artifacts and noise in recording renders many beats (approx. 300 million of 1.2 billion) invalid. To preserve as much source data as possible, we evaluated the validity of each beat using the signal quality metric defined in [2] and beatDB records this validity *per beat*.

We identified at least 120 beat features that will be frequently used and which are relatively quick to compute: *features with medical relevance* like diastolic, systolic pressures and beat duration;

time domain features like mean; and *morphologic* features - distance between two subsequent beats and features extracted from frequency domain. beatDB will eventually hold these pre-computed features per beat at an approximate storage cost of approximately 8GB per feature. The database is organized along patientID, indices for start time and stop time for a beat and the features. There are currently approximately 1.2 billion beats from approximately 6000 patients and 18 features. The beatDB feature extraction engine runs in parallel on hundreds of nodes on a private cloud, extracting features for one patient at a time and populating our database. This data is accompanied by software that allow compositions of data to support multi-scale, multi-representational learning. This software has strategic parameterizations which expose data processing options and facilitates agile distributed computation.

Joint-time frequency domain features like wavelets require two different approaches because for each beat a wavelet generates a large matrix of scale and shift coefficients. A specific scale and shift parameter set can be extracted or a "service" yet to be implemented) will employ a search and score methodology to identify the best possible set of wavelet coefficients for the problem at hand by using a user defined metric. For example, for a classification problem cross validation accuracy would be used to rank the quality of a set of candidate wavelet coefficients.

BeatDB has sub-frameworks which support knowledge discovery. The **analytics** sub-framework provides descriptive statistics such as how many patients had an acute hypotensive event, information content of patients' segments (measured via compressibility) and patient segment quality. In the **hypothesis testing** sub-framework the clinician posits a hypothesis that distribution of an assembly of data under a certain condition (case) is in fact statistically different than its distribution when the condition is not present (control). It is described further in Section 3. Two other sub-frameworks are pending integration. The **clustering** sub-framework will use unsupervised learning to cluster the patient time series while establishing cluster labels. It will allow a clinician to request knowledge about a patient who is similar to a cluster. **Multi-representational learning** will support a number of machine learning algorithm like dynamic Bayesian networks and several classifier algorithms. We also plan a sub-framework for **crowd sourced feature discovery**. Researchers will be able propose a new beat feature and submit a script that extracts it from a beat. Without computing this feature for the entire DB of beats, beatDB will estimate its correlation with existing features via bootstrapping. This will inform the value of full extraction.

3 Hypothesis Testing Framework

BeatDB's hypothesis testing framework permits researchers to posit and test whether there is discriminatory power in the data with regard to a condition, or whether there any predictive precursors to it. The condition may be externally defined, i.e. via clinical data, or be detectable within a blood pressure waveform. To accommodate a "hunch" or inexact notion, the framework allows a hypothesis to be imprecisely expressed. This is accomplished by parameterizing how a condition is defined, how covariates are aggregated and how prediction is defined (lead, lag, features). In detail:

Step 1: Define condition The researcher selects patients known to have experienced the condition and some controls. For example, an acute hypotensive event defined by mean arterial pressure could be of interest. BeatDB then uses a software *scanner* script, composed by the researcher, to scan segments and identify the condition's start i and stop j time indices, where present. The condition's definition is parameterized per Table 1. The pre-condition part of the segment, up to the start time index, $f(t = 1 \ldots i - 1)$ becomes the signal which is examined for precursors or tested for discriminatory power.

Step 2: Define data aggregations: The pre-condition part of the segment is divided into non-overlapping windows and any number of covariates, e.g. different moments or trend, are formed. The researcher selects covariates parameterizations per Table 1.

Step 3: Define prediction assumptions: The researcher chooses the *lead* and *lag* for the prediction problem and selects a machine learning algorithm, e.g decision trees, SVM, logistic regression.

Step 4: Choose hypothesis evaluation metrics: The researcher selects an evaluation metric e.g. area under the curve, Bayesian risk for a given cost matrix or Neyman Pearson criterion.

BeatDB next sweeps the combined ranges of the parameters experimentally. For each parameter set, it returns a result, according to the researcher selected metric so that a precise (e.g. the strongest exact) hypothesis can be identified.

Parameter class	Parameter names
Condition definition parameters	Window size, threshold, frequency, variable
Prediction parameters	Lag, lead, features
Data aggregation parameters	Sub-aggregation window, sub-aggregation function, aggregation window, aggregation functions

Table 1: Hypothesis testing framework parameterization

3.1 An Example: Discriminating Acute Hypotensive Events

We demonstrate the framework with a hypotension condition. An acute hypotensive event (AHE) is defined in terms of a **variable** when, for some condition **window,**, with a **minimum frequency (%-age)**, blood pressure dips below a threshold (in mmHg). For example, 90% of average MAP values in a 30 minute window dip below 60 mmHg. A clinician is willing to select a precise definition of the threshold on the basis on discriminatory information. The framework can vary this threshold (here between $[50, 80]$) and, per Figure 1, provide useful information on the resulting data. Here, it indicates how many unique patients experienced AHE, how many unique AHE cases were identified and how the case to control ratio changed. Per Table 2, a clinician then defines values for the data aggregation parameters and value ranges for prediction lead and lag.

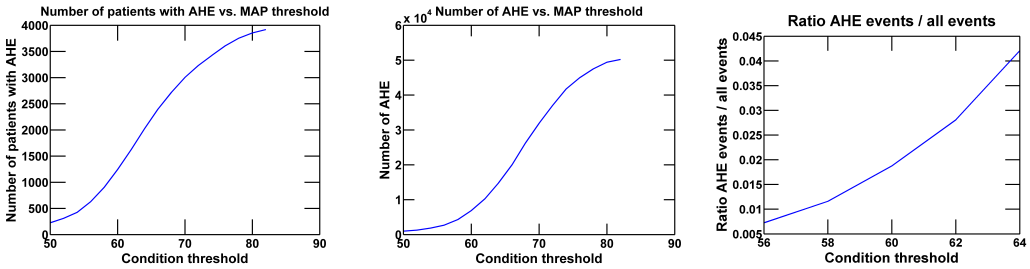

Figure 1: (Left): Number of patients with AHE cases as we change the threshold. (Center) Total number of AHE cases present as we change the threshold for the condition. (Right): Balance between the AHE and non AHE cases as the thresholds are varied.

Parameter names	Parameter choice
Condition definition: window duration	30 minutes
Condition definition: variable	minute average MAP (mean arterial pressure)
Condition definition: variable threshold	[56, 58, 60, 62, 64] mmHg
Condition definition: frequency	90% minimum
Prediction definition: lag	[10, 20, 30, 60] minutes
Prediction definition: lead	[10, 20, 30, 60, 120, 180] minutes
Prediction definition: covariates	14 covariates
Sub-aggregation window	1 minute
Sub-aggregation function	Mean
Aggregation window	Prediction definition: lag
Aggregation functions	mean, standard deviation, kurtosis, skew, trend
Machine learning algorithm	logistic regression
Hypothesis evaluation metric	AUC (area under the ROC curve)

Table 2: Parameter Sweep for AHE Discrimination. The 14 covariates are: root-mean-square, kurtosis, skewness, systolic blood pressure, diastolic blood pressure, pulse pressure, beat duration, systole duration, diastole duration, pressure area during systole, standard deviation, crest factor, mean, mean arterial pressure.

The framework then applied the 5 different aggregate functions to 14 different per beat features, resulting in 70 features per training exemplar. Then we prepared a dataset for each parameter combination (5 MAP thresholds, 4 lags and 6 leads, resulting in 120 combinations). On these, we

executed 10-fold cross validation on our lab's private cloud using approximately 2 nodes with 24 VCPUs each for 48 hours. The results across different condition thresholds and prediction leads and lags can be viewed in Figures 2 and 3.

3.2 Results and Discussion

Figure 2 (left) shows the area under the ROC curve (AUC) for different lead times, given maximum lag of 60 minutes for 5 different MAP thresholds. The area under the curve drops as lead time increases. The prediction problem becomes easiest when a threshold of 56 mmHg is applied to average MAP. As this threshold increases, predicting AHE, given this data, becomes harder. This confirms expectations because 56 mmHg is an extreme threshold point and we would expect that the cohort of patients with such an extreme condition will be significantly be different then the rest of the patients. Figure 2(right) shows the AUC when we change the lag and keep the $lead$ at its minimum 10 minute duration. In this case, we see that the performance improves as we increase the lag or historical data taken into account. This is intuitive because a longer lag provides more signal to learn from.

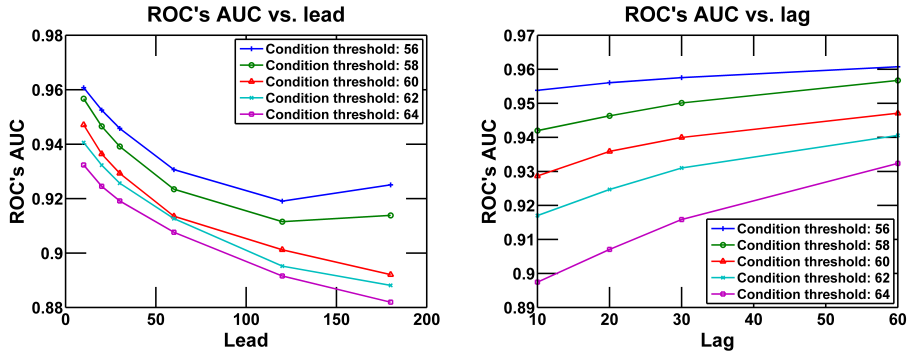

Figure 2: Discriminatory strength in terms of Area Under the Curve for the classifiers for different condition threshold with prediction lead and lag sweeps.

Next, in order to explore a point solution on the ROC curve, we chose a true positive rate of 90% for AHE and evaluated the false positive rate. Figure 3 shows the results for different lags and leads for different definitions of AHE. Here, again, more signal history helps prediction.

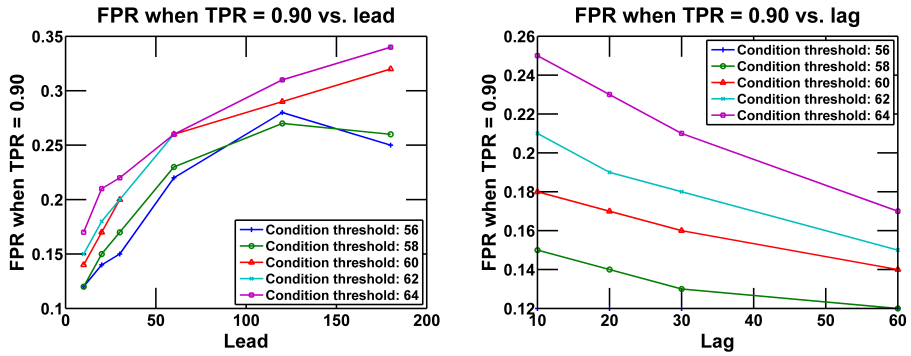

Figure 3: False positive rates achieved for a true positive of 90% as we changed the prediction problem parameters and the thresholds for AHE determination.

References

[1] 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.

[2] 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.