Feasibility Document

11/13/2017

Dr. Fatemah Afghah

James Todd, Nathan Payton-McCauslin,

Alexander Grzesiak, and Robert Rasmussen

Data Menders

Table of Contents

1.0 Introduction

Data Menders is a capstone team aimed to help facilitate the reduction of false alarms in intensive care units (ICUs) through the analysis and interpretation of signals to determine the true nature of an alarm. Our client and sponsor, Dr. Fatemeh Afghah, is an assistant professor in the School of Informatics, Computing, and Cyber Systems at Northern Arizona University. It is through real life experiences that this problem as come to fruition as a capstone project. In a typical ICU, nurses are constantly scrambling tending to fires that may not exist. ECG monitor, patients move, muscles have spasms, patients have a life-threatening condition…these are all causes of the same alarm that force a nurse to respond to a patient but as you can tell the latter deserves the most urgency. This is where a piece of software come in to analyze multiple signals from a patient and determine what kind of care is called for. Each year a number of patients die waiting for nurses to finish responding to false alarms, in fact, from 2005 to 2008, the FDA database received 566 reports of patient deaths related to alarms of monitoring devices. Our software will hopefully alleviate the stress put on ICU medical staff and as a result save lives.

This document serves to outline the foreseeable technological hurdles that might arise throughout our project. Section 1 briefly covers the challenges we expect to come up. Section 2 elaborates on the broad challenges defined in Section 1 with an introduction of the challenge, the possible solutions to the problem, the solution that seems the most logical at the moment, and lastly some preliminary research and logic that shows feasibility.

2.0 Technological Challenges

This section aims to outline the main technological challenges that we anticipate going forward. This is by no means an exhaustive list as there are bound to be others that come up. However, with the knowledge we have about the state of affairs and where we see the project going, these are the challenges we foresee.

1. We will need to decide on the best programming language to use.
2. We will need to determine the best transform method for a condition.
3. We will need to denoise the signals before analyzing them.
4. We will need to identify key features in the signals.
5. We will need a method of testing the analysis while a signal is flowing.
6. We will eventually need a way to store patient's previous ECG signatures to test future signals historically normal events don’t trigger alarms.

3.0 Technological Analysis

In this section the challenges are expanded upon. Each issue contains an introduction that elaborates on the specifics of the problem and what exactly makes it an issue for us going forward. This is followed by the possible alternatives that make the most sense at this time detailing the pros and cons of each. Once the possibilities have been outlined and discussed, the one that we will be using as well as the reasoning or proof behind the choice will be made clear.

3.1 Deciding On a Language

Whatever language is decided on has to offer exceptional math library support considering the vast amounts of signal processing that will take place. Importing the data will be relatively straight forward for any language but transforming the raw signal using various transform methods will require utilizing many libraries as these methods will be cumbersome not to mention redundant if we were to write these. This is one of the biggest factors that a language must have but another key feature is going to be how the language responds to large data sets. ECG datasets can get on the order of millions of entries so efficiency is key especially considering this analysis has to be done in real time if in a real world environment.

The two contenders for programming language are MatLab for its vast assortment of built in tools and C/C++ for it's low level speed. These two each have exactly one key feature that we are looking for so it comes down to which feature do we value above the other.

MatLab makes it incredibly easy to write and test code that is heavily dependant on very math intensive operations such as transforming a signal from time to frequency domain and back to time again. Importing data, transforming the data, and plotting it takes 3 lines of code which goes to show how much it lives up to its name. The place that MatLab starts to break down is when the dataset gets large and when trying to design a stand alone program. There is so much MatLab is trying to do behind the scenes that time starts to play a role when analyzing data. Importing data is extraordinarily easy but the cost here is it can take just as long to import an ECG signal as it did to create it. Beyond this issue, MatLab is able to create a basic UI and even compile to an executable in order to run on a machine that doesn't have MatLab installed. Another option to create a standalone application is to have MatLab convert the code into C/C++ code which could be very useful if we run into the problem highlighted earlier regarding speed degradation when dealing with large datasets.

C/C++ on the other hand shifts work away from built in operations and onto the programmer. Luckily there are still libraries that can be linked in order to get a similar experience to that of MatLab. These libraries may be a bit more cumbersome than the simplistic nature of MatLab but they are much faster than even the most optimized built in transform function in MatLab. One last factor that has to be addressed is the increased programming time that would occur when using C++ rather than MatLab. Programming time is a resource just like run time is so this will have to be managed accordingly.

Given the complex mathematical nature of the work we are doing, the language we will be starting with is MatLab. The simplicity makes it attractive while designing the algorithm that will guide our program. That being said, we do acknowledge the obvious benefits that C/C++ offers so once the logic side of the program is ready, we will make the switch to C++ most likely. There is such a steep learning curve with this material that struggling with the content on top of dealing with some of the nuances of transforming the data using C++ makes it unappealing to start off in anything but Matlab. The majority of time spent will be researching the various methods of analyzing the data in order to try to detect heart irregularities with a smaller portion coming in the form of actually coding so the time spent in MatLab getting the algorithm down will be invaluable before we switch to implementation in C++.

3.2 Determining A Transform

Before denoising or analyzing the raw signal, it must first go through a process to transform it into a domain that gives more useful information than than the raw signal. There are various methods we can implement to move the signal into a different domain with each having its own list of pros and cons. The main transforms we will be using are those based on the Fourier and Wavelet transforms. There are many variations so below are details for these two categories of transforms.

Fourier transforms take some input signal and outputs magnitudes for various frequencies that occur in the signal. The data coming out of fourier transforms can tell you with certainty which frequencies were present in the signal. This can be incredibly useful in isolating certain frequencies in the signal but as a result of integrating over the entire signal, all time awareness is lost. There are methods than attempt to work around this by breaking the signal up into smaller pieces and doing fourier analysis on the smaller sections.

The Wavelet class of transforms can be thought of this this way. In order to give frequency and time information about a signal, wavelet integrates over the entire signal just like fourier but with respect to where the frequency happened. A consequence of this is there is a loss in precision when it comes to the frequency as well as exactly where it happened. This loss in accuracy can be mitigated with windowing techniques and correct wavelet choices but in practical cases is not usually a major issue.

3.3 Reducing Signal Noise

Reducing signal noise is important to getting a cleaner, more accurate and as a result more usable signals. Some of the denoising algorithms that will be used in this project are Wavelet transform and windowing of a discrete fourier transform, Gaussian transform.

Something to keep in mind when denoising is that we do not want to lose data. That being said, denoising is still an important process for us to do. Some common causes of noise in a medical environment are as follows: talking, moving, power cables, touching the sensors that is recording the signal, etc. Denoising is important because noise could trick us in believing there is a problem when there is not, and over all it can give better trends in the data. This is dependant on the signal but generally some level of denoising needs to be done before the signal can be further analyzed. One of the main parts to this project is understanding what signal analysis to do on a given signal and to give these algorithms the best chance of doing their job, we have to feed them information that is clean and ready to use.

The problem is, after denoising, did we lose some data that was crucial in showing a heart condition or more seriously, did we lose data that showed that the patient did in fact have a heart condition that we needed to catch. This can be mitigated by only doing the minimum amount of denoising that is necessary in creating a usable signal but still contains all the detail of the signal that we care about.

We will use all techniques to gather data, then with that data, we will choose which method gives the stronger result. Or in other terms, to which level should we denoise the signal? This is a question that will have to be answered based on the specifics of the signal that comes up.

3.4 Identifying Features
This is one of the more challenging aspects of this project. A patient hooked up to an ECG machine will give a signal but this signal means nothing unless we can take it and be able to tell if something is wrong with a patient. Denoising the signal mentioned above is the first step in uncovering any irregularities in a signal. The next is using various mathematical equations and a bit of ingenuity to transform the data that can then be analyzed to pinpoint specific features to help in determining what is going on with a patient.

Once a signal is transformed there are a couple trivial features that can be extracted. Features such as mean, median, mode, max, min, range, variance, and standard deviation are those that can pretty easily be taken from a set of data. Others such as those derived from game theory algorithms or training a neural net to extract features are also an option however are much more involved.

As for which features we will be using, if there is a feature that can be implemented we will use it. Starting off, the basic mean, median, mode, and equivalent will be the features that will be used for preliminary testing with more to follow. There is no single feature that can give an indication of when something is going wrong so it will be paramount to have a large set of features that are calculated. This will be a topic we will do more research into which will hopefully produce a set of features that offers the best insight for a particular heart condition.

3.5 Test Algorithm Using Real Data

When it comes to how we are going to test our program and the algorithm behind it, there is a need for data and lots of it. It is impossible to write an automated test that determines whether or not comparing a certain feature works in figuring out someone has a heart condition. To add to this there is a need for a very specific set of data to train any potential neural net or other machine learning algorithm.

We will be testing our algorithm using data obtained from real patients. We will have two sources of data: one from online databases containing ECG signals along with metadata about the conditions that are present in the signal, and one from our own vitals. The online data set contains readings from patients with health problems as well as healthy patients. This is attractive to us because it offers an opportunity to test our solution against patients whom we expect there to be no alarms. Along with this data pulled from online, we will also be using an ECG in a lab setting to obtain more, hopefully, healthy data.

Another area to focus on is how the program will respond to live data. Using a live connection will present its own set of problems that testing on fixed length doesn't provide. Because of this, using the ECG available to us will give us the possibility to test this in a real world environment with similar signal corrupting environmental factors.

There is no single source of ECG data we are looking for. Each signal offers its own noise signature and set of circumstances. In order for us and our program to be able to successfully identify heart conditions in any sort of signal, it is very important in the long term viability of this software that we take as many test signals as possible.

3.6 Storing and Using Patient Data

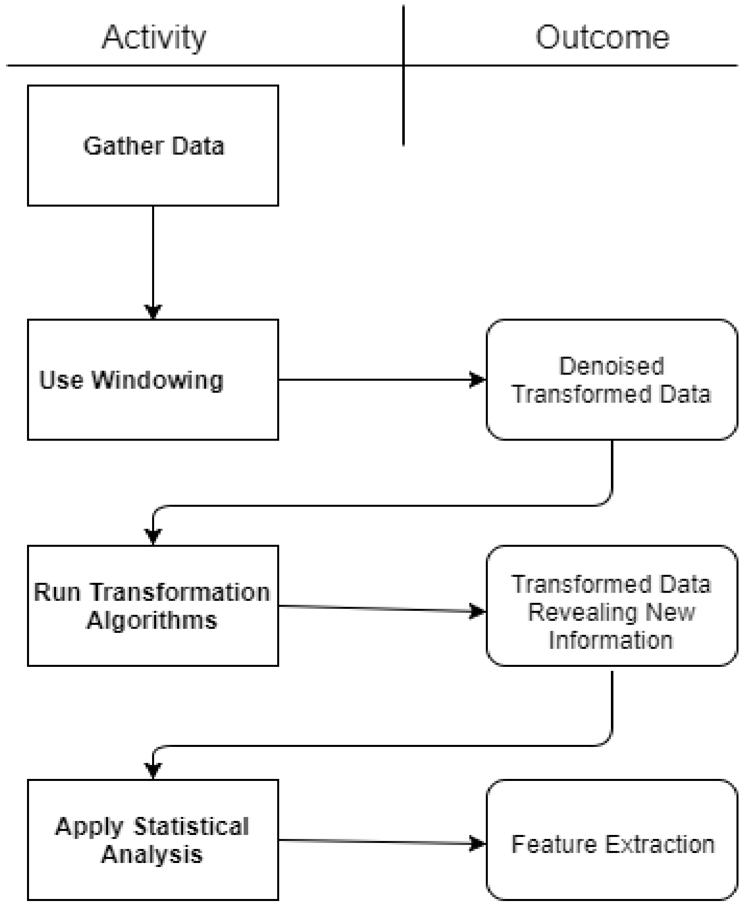
An issue with some patients is their heart conditions may be preexisting and historically non-life threatening which means an alarm for them might not necessarily mean something is wrong yet most indications might point that way. This can be problematic because more than likely, an alarm is going to sound sending a nurse scrambling only to find out that Bob had a heart murmur for the 45th time in a day. A potential problem that could arise here is the opportunity for a false negative. If our patient, Bob, were to actually have a heart condition that was serious, the algorithm should not be so selective that it mistakes a true heart condition that needs medical attention for a common occurrence.

Once features can successfully be analyzed and produce recurring correct results we can start to take the patient's own previous vital signs into account as an added layer of analysis. This can be done by taking a set of a patient’s data and using it as the “function” by which we compare the live signal to in order to scan for new and dangerous abnormalities. This is the abstract model of how a program could use machine learning to come to a conclusion. There is an assortment of methods in which this can be done. The Pan-Tompkins algorithm uses the highest point in an ECG signal (R point) to detect arrhythmias, statistical approaches, fuzzy set theory that aims to model the randomness in humanity but suffers from subjectivity, as well as differing neural network approaches.

At this point there is no definitive solution but rather a set of solutions to a larger set of problems. As of now, there is no conclusion we can draw to this topic but will instead be a continuing point of research. Like with feature selection, there is no one-size-fits-all solution. Each solution offers its own benefits and drawbacks but at this time we are in no position to offer a method of giving out program a method to “think” for itself given the lack of context and facts surrounding the decision.

4.0 Technology Integration

In the last section we outlined challenges in our project and how to solve them. These challenges had to do with transforming, denoising and analysing data as well as extracting features out of that data. In this section we will explain how all of these smaller solutions come together to create the overall system. The way the smaller solutions fit together is outlined in the diagram below. The left-most straight edge boxes indicate the technique used while the rightmost rounded boxes indicate the outcome of said technique.



Activity
Outcome
Gather Data
Use Windowing
Denoised Transformed Data
Run Transformation Algorithms
Transformed Data Revealing New Information
Apply Statistical Analysis
Feature Extraction

*note that the data gathered may be a sample or stored real data about the patient.

The diagram shows the general flow and how the solutions fit and interact with each other but it does not detail the language we will be using. This is because as mentioned in the last section, code is written in MatLab to prototype and research but it is then converted over to C/C++ to increase the speed of program execution. For greater explanations of each technique performed refer to the corresponding subsections in section 3.

5.0 Conclusion

Team Data Menders believes that, by using current technologies and publically available data, our goal of analyzing medical data to more accurately determine issues is feasible. By starting with Matlab, we have the ability to quickly prototype algorithms to find a good match. By using Fourier and Wavelet transformations, we gain the ability to denoise and analyze the signals. From there, we can extract features that indicate problems with the patient. Because we have both healthy and unhealthy data to test with, we can perform tests on both sets of data to see how well our algorithms function. Considering the above, we believe that our project can be completed by Spring of 2018.