Classification of Non-Invasive Fetal ECGs with Dynamic Time Warping

Kellin Rumsey
Department of Mathematics and Statistics
University of New Mexico
Data Mining - Final Project

I. Introduction

With modern medical technology, heart rhythm can be easily, safely and accurately read using an electrocardiogram (ECG). Medical professionals can detect problems and irregularities, both big and small, effectively, sometimes with the help of computer aided pattern recognition. The difficulty of the problem however drastically increases when the patient of interest is not a fully grown adult, but an unborn human child. While accurate invasive ECGs exist, they are seldom used (except perhaps at the time of pregnancy) due to the health risks and high cost. Thus a popular technique known as *Non-invasive Fetal ECG* (NI-FECG) is commonly used.

In NI-FECG, a series of sensors are placed on the mothers abdomen, and are used to detect the fetal heartbeat. The drawback of this safe and easy alternative is a noisy signal, due to the muscles in the abdomen and the unavoidable fact that the maternal heartbeat is dominant. Data mining problems such as event classification, abnormality detection and FECG extraction are clearly important. Data mining researches have had some successes in recent years with these problems, but limited data forms a bottleneck. The *Fetal ECG Synthetic Database* was created to address this data bottleneck, using the FECGSYN simulator to construct a wide variety of scenarios. Specifically, the data is composed of 1750 five minute ECG samples each with 5100 hours of data and 1.1 million fetal ECG peaks.

In this paper, we examined 15 simulated pregnancies, using 2 minutes of FECG data in each pregnancy. The pregnancies were divided into 3 classes.

- **Class 1:** Normal pregnancy
- **Class 2:** Ectopic heartbeats
- **Class 3:** Twin pregnancy

Normal pregnancy is chosen as a baseline case, and twin pregnancy was chosen for interpretability of ECG data. Ectopic heartbeats refer to a disturbance in cardiac rhythm and is a type of arrhythmia. While it can be common and un-alarming in adults, the presence of ectopic beats in a baby can be an indicator of potentially serious heart disease. Thus classification of these ectopic beats is an important and challenging problem. Figure 2 gives a deconstructed view of the signals in discussion. Specifically, these signals represent a two second snippet of the ECG for a twin-pregnancy. The top three panels show (from left to right) the ECG corresponding to the mother and each fetus. It is worth noting that the magnitude of the Fetal ECGs is comparable to that of the Noise signals and much smaller than the magnitude of the healthy MECG. Additionally, the noise level varies for each pregnancy, and the pregnancy shown in Figure 2 is relatively well-behaved comparatively. These facts combine to make this a challenging problem.

Data Collection and Processing

The FECGSYNDB Database is hosted at https://physionet.org/physiobank/database/fecgsyndb/. Databases on Physionet require a special tool which must be downloaded, and a terminal window is used to convert these files to lengthy text files. Although the data is too large to be held in memory (on my laptop at least), I extracted the majority of the dataset for purposes of visualization and to help hone in on a reasonable problem. Once the classification task was laid out, the following steps were taken one pregnancy at a time.

1) Convert the data to a text file.
2) Read the text file and extract 2 minutes of ECG data from the first channel.
3) Extract as many maternal heart beats as possible.
4) Regularize the data. This included applying low and high pass Butterworth filters, z-normalizing each subsequence

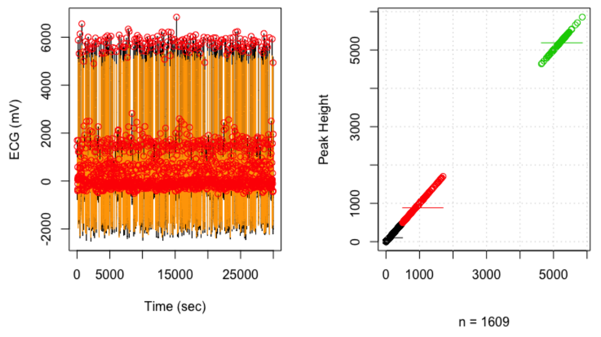

Fig. 1. Primary peak extraction process. Left panel shows filtered data with all identified peaks. Right panel demonstrates clustering to select only the primary peaks.

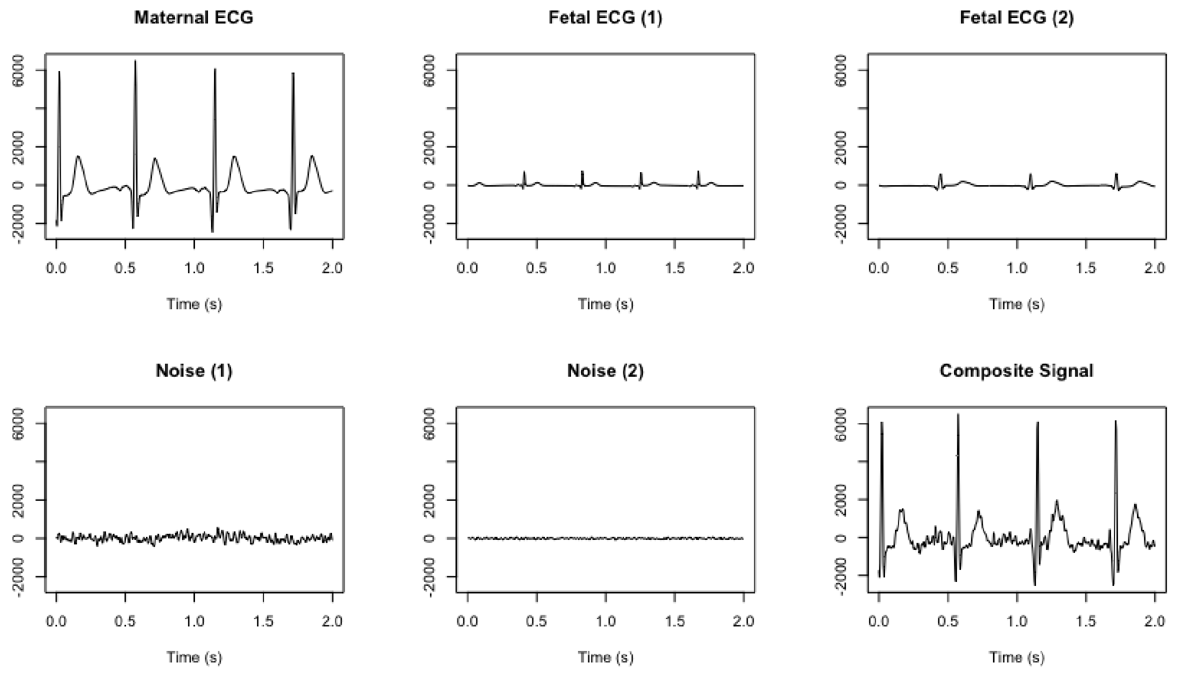

Fig. 2. Decomposition of signals. The observed signal (bottom right) is the sum of the 5 source signals. These signals represent a twin pregnancy. Normal and ectopic classes have only 1 FECG signal.

and scaling/interpolating so that each beat is takes place over a standardized time between 0 and 1.

Most steps of the collection/processing stage are straightforward (although time consuming). The maternal hear beat extraction was tricky, as the author was (and is) unaware of a gold-standard which was within the scope of this class. So a relatively simple and fairly effective method was employed. By filtering the data, we are able to easily detect all spikes above a certain threshold. This includes the MECG spikes, as desired, but also contains a large number of undesirable spikes due to fetal heartbeats and noise. The height of each peak was then collected, and a efficient implementation of the univariate k-means algorithm was employed. Assuming a normal probability model, the number of classes was selected using Bayesian information criteria (BIC). The group with the largest mean was deemed to be the MECG spikes which we were looking for. A summary of this process is shown in Figure 1.

In some cases however, the noise level was so large that a high-peak class couldn't be accurately detected without several false positives or false negatives. To handle this, we simply removed all "heartbeats" whose duration in seconds was deemed an outlier according to the $1.5 \times IQR$ Rule. Specifically, given a set of heartbeat durations $x_1, x_2, \cdots x_n$, we respectively denote Q_1 and Q_3 to be the first quartile and third quartile of the x_i and define the Interquartile range $IQR = Q_3 - Q_1$. The upper and lower fence are thus given by:

$$LF = Q_1 - 1.5 \times IQR \qquad UF = Q_3 + 1.5 \times IQR$$

Thus any heartbeat duration x_i such that $x_i < LF$ or $x_i > UF$ was denoted a "false heartbeat" and was eliminated. In the end, we collected 1600 subsequences of length 500. An illustrative sample of these z-normalized heartbeats is shown in Figure 3. The dashed black line in each panel represents the DTW Barycenter Average (DBA) of the set. The dotted red line

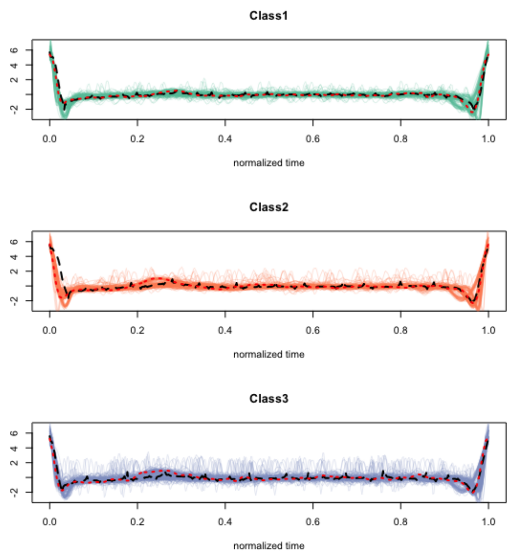

Fig. 3. Representative heartbeats by class. Dashed black line represents DTW Barycenter Average (DBA). Dotted red line is Extremal Depth median function.

corresponds to the median function according by Functional Extremal Depth. [1]

II. Classification Algorithms

After normalizing the heartbeat data, the goal is to classify the heart beats in one of three classes: (1) normal pregnancy, (2) ectopic fetal heartbeats or (3) twin pregnancy. The data was split into 5 approximately equal sized groups. The first set of functions was held out as a validation set V. The remaining 4 groups were then used in a simple 4-Fold Cross Validation analysis.

For nearly any classification algorithm, we need a notion of distance. Since interpolation was used to give each time series equal length, the *euclidean distance* metric is the first possible option. That is, for two series $x = (x_1, x_2, \cdots x_m)$ and $y = (y_1, y_2, \cdots y_m)$ we define

$$d_e(x,y) = \sqrt{\sum_{i=1}^{m}(x_i - y_i)^2}$$

Another option is to use *Dynamic Time Warping distance*, which is defined recursively. DTW distance is defined recursively as follows. We start with the base cases:

$$\gamma(i,j) = \begin{cases} 0 & i = j = 0 \\ \infty & i = 0 \text{ xor } j = 0 \end{cases}$$

Then starting at cell $\gamma(1,1)$ we recursively compute

$$\gamma(i,j) = (x_i - y_j)^2 + \min[\gamma(i-1, j-1),\ \gamma(i-1,j),\ \gamma(i,j-1)]$$

Finally, we define $DTW(x,y) = \sqrt{\gamma(n_x, n_y)}$ where n_x and n_y are the length of x and y respectively. DTW distance, unlike it's euclidean counterpart, has the ability to recognize when two sequences are really very similar, while possibly out of alignment. It also has the ability to provide a notion of similarity when $n_x \neq n_y$. where euclidean distance fails. There are two potential drawbacks to DTW distance however. The first, is the complexity, which (assuming $n = n_x = n_y$) is $\mathcal{O}(n^2)$ compared to $\mathcal{O}(n)$ for the euclidean measure. Second, DTW distance is sometimes *too flexible*, and can allow two sequences which should be considered dissimilar to have small distance. We can solve both of these by adding a warping window parameter $w \in \{1, 2, \cdots min(n_x, n_y)\}$. The definition of DTW distance can remain the same, with a small change. The base case becomes

$$\gamma_w(i,j) = \begin{cases} 0 & i = j = 0 \\ \infty & i = 0 \text{ xor } j = 0 \\ \infty & |i-j| > w \end{cases}$$

[1]Functional ED is a statistically rigorous framework for performing non-parametric statistics on functional data. We originally intended to apply some of the methods to this classification task, but the algorithms scale poorly in both the number of functions and number of points, and the idea was abandoned.

This puts a limit on how far a function is allowed to warp, while simultaneously reducing the cost to $\mathcal{O}(wn)$. It is easy to see that when $n_x = n_y$, setting $w = 1$ implies that $DTW_w(x,y) = d_e(x,y)$.

A. A Simple Partition Based Classifier

To start things off, we propose a simple partition based classifier similar in spirit to LDA or K-means. To do this, we need a measure of center. It is well known that arithmetic averages of time-series can be misleading and a poor choice of center in the presence of phase mis-alignment. Thus we turn to the DTW Barycenter Average (DBA) which is found iteratively and tries to minimize the distance from each function in a set with respect to DTW_w. An alternative approach is to use the a median function based on the notion of Extremal depth, which would allow for a partitioning approach similar to that of PAM, but it is too currently too expensive for this application. Thus our algorithm can be written as follows:

```
ALGORITHM ONE:

S = training set
T = test set
Construct Sk = {x in S | x has class k}
Compute DBA(Sk)

For each y in T:
predicted class of y is argmin_k(DTW_w(y,
DBA(Sk))
```

B. Early Abandoning and Lower Bounding

This algorithm is fast because it requires at most Kn_y distance calculations (where K is the number of classes) once the DBA's have been found. As we will shortly see, early abandoning and a cascade of lower bounds can reduce this number even further. While this can speed up the partition-based method slightly, these methods will help tremendously for our final classifier.

In `Algorithm One`, we are trying to find z in a small set Z that minimizes $DTW_w(y,z)$. One way to produce a small speedup is via early abandoning. There are some highly sophisticated techniques, but a fairly simple implementation was used here. We take advantage of the following simple fact. For all i, $\min_j(\gamma_w(i,j)) < DTW_w$. Thus if we keep track of the current minimal distance between y and z, we can abandon our calculation of DTW as soon as the smallest element in a row of γ exceeds the best distance so far. In some cases, this can happen very quickly leading to a drastic reduction in computation time.

Lower bounding refers to any function such that $LB(x,y) < DTW_w(x,y)$, where ideally $LB(x,y)$ is much faster to compute that DTW distance. Now if the current smallest distance

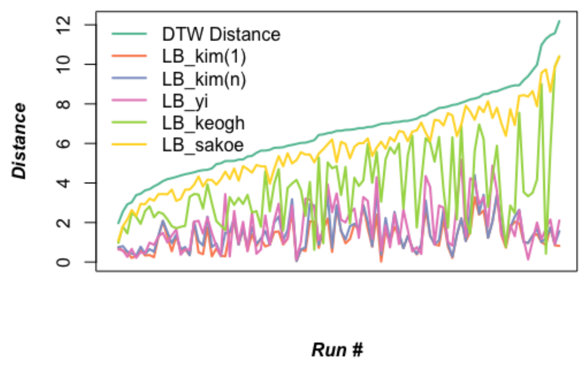

Fig. 4. DTW and 5 lower bounding functions for 100 randomly selected pairs of heartbeats.

is δ, we can check $LB(x, y) > \delta$. If this is true, then there is no need to compute the distance between x and y. This can lead to us calculating a very small portion of DTW distances. There is typically a tradeoff in LB distances between tightness of the bound and time to compute. The idea behind cascading is to compute the cheapest bound first and check the condition. If it fails, then compute a more expensive, but hopefully tighter upper bound. Compare and repeat until we either find $LB(x, y) > \delta$ and we skip the calculation, or until we run out of of lower bounding functions and are forced to compute DTW distance. In our implementation, we use the following sequence of lower bounds.

- $LB_{kim_1}(x, y)$. Euclidean distance between the vectors (x_1, x_n) and (y_1, y_n). Very cheap. Took 0.1 seconds to compute 10,000 times.
- $LB_{kim_2}(x, y)$. Euclidean distance between the vectors $(x_1, x_n, x_{(1)}, x_{(n)})$ and $(y_1, y_n, y_{(1)}, y_{(n)})$. Where $x_{(1)}$ and $x_{(n)}$ represent the min and max of x respectively. If $w < n$, we need to also make sure than the min (and max) of x and y are within w time points of each other or this isn't actually a lower bound. This is $O(n)$ to compute. Took 2 seconds to compute 10,000 times.
- $LB_{yi}(x, y)$. Creates a horizontal band around x. Take euclidean distance between this band and points of y which lie outside the band. Repeat by swapping the roles of x and y take the max. Also $O(n)$ but with bigger constants. Took 3 seconds to compute $10,000$ times.
- $LB_{keogh}(x, y)$. A little bit more expensive, but effective. Similar to LB_yi but uses a sliding window to construct band rather than horizontal. Took 15 seconds to compute 10,000 times.
- $LB_{sakoe}(x, y)$. Similar to LB_keogh but with a different band which produces a tighter bound. Took 16 seconds to compute $10{,}000$ times
- DTW distance. Tightest lower bound possible. (: Took 41 minutes to compute $100{,}000$ times.

All of these LB functions were implemented manually in R, except for the Sakoe-Chiba bound which I found in a package. Figure 4 illustrates the tightness of the bounds for 100 randomly selected pairs of heartbeats.

C. *The 1 Nearest-Neighbor DTW Classifier*

This algorithm can be considered Hierarchical (especially if we were clustering), but is similar in flavor to the partitioning method described above. Rather than finding the nearest centroid however, we need to search through the entire training set to find the nearest neighbor of x and we predict that x has the same class as its nearest neighbor. This runtime will clearly have complexity $O(n_{test} n_{train})$ instead of $O(K n_{test})$, but the early abandoning and lower bounding can be far more effective here leading to an algorithm with a linear amortized cost.

```
ALGORITHM TWO:

S = training set
T = test set

for each x in T
find y in S such that DTW_w(x, y) <
DTW_w(x, z) for all z in S
predicted class of x is class of y
```

Here it is very important that we utilize early abandoning and lower bounding. As we will see, it can drastically reduce the number of DTW calculations. As a final speed up, we suggest the following. Lower bounding works best if the best distance so far is small. If we have already run the much cheaper partitioning based method, we can order the appearance of the $y \in S$ so that we are likely to find the nearest neighbor earlier. This doesn't change the complexity of the algorithm, but over many runs it can reduce the constants.

III. Results

One of the first problems we must consider is the selection of w in the DTW distance function. Figure 5 shows the $DTW_w(x, y)$ as a function of w for a randomly selected pair x and y. Although there is variability based on the pair selected,

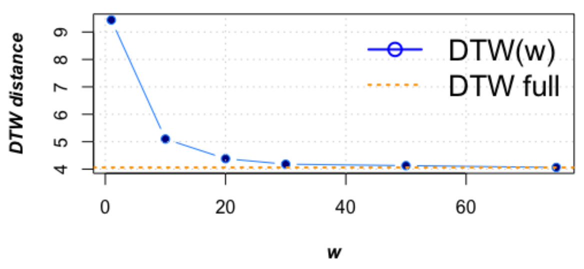

Fig. 5. DTW distance as a function of w. Dotted orange line is DTW for $w = n$

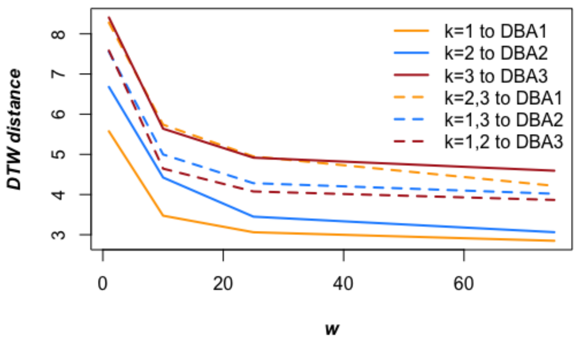

Fig. 6. Average distance to barycenters for subsequences in different classes.

this was fairly representative of the effect. It is clear that for this heartbeat data, setting $w = n$ is overkill. In fact based on Figure **??** a value of $w \approx 25$ seems to achieve close to the same value as the full DTW. In addition, we used a small subsample to approximate within-cluster distances and between-cluster distances using class DBA's as a proxy. These results can be seen in Figure 6. We might hypothesize from Figure 6 that the first class will be the easiest to predict, and the third class the hardest.
Using 4-fold Cross validation, we consider four values of w. The partition based classifier illustrates mediocre performance with a maximum accuracy of about 60%. The algorithm is fast though, as promised, and we have leveraged its speed to give a boost to the 1-NN DTW algorithm as well.

TABLE I
CROSS VALIDATION RESULTS FOR PARTITIONING METHOD

w	Accuracy	Avg Prec.	Avg Recall	Avg F1-score	Runtime (s)
10	57.26	57.73	62.69	57.73	22
25	57.76	57.82	60.47	57.82	35
50	59.23	61.1	62.55	61.1	47
75	58.31	59.14	65.96	59.14	60

The single nearest neighbor classifier with DTW performs significantly better, obtaining accuracies over 97%. The algorithm doesn't scale particularly well as w increases however, and the marginal gains from increasing w seem to be lost beyond $w = 50$. Thus a value of w between 25 and 50 seems like a reasonable choice here. The final column in Table 2 gives the fraction of DTW calculations which were required. When w is small, only a small fraction of the calculations are needed, but the lower bounding becomes less effective when w is increased.

Figure 7 shows the runtimes as a function of n for randomly selected subsets S and T of size n. We can see that the early abandoning and lower bounding makes the amortized complexity linear as promised. The re-ordering also helps, but just by a little bit. Although it is possible that the benefit of reordering grows with the number of functions.

TABLE II
CROSS VALIDATION RESULTS FOR 1-NN DTW METHOD

w	Accuracy	Avg Prec.	Avg Recall	Avg F1-score	Runtime	DTW %
10	92.66	92.9	92.83	92.86	144	2.5
25	96.91	96.94	96.91	96.93	402	4.4
50	97.1	96.84	96.92	96.88	1302	7.2
75	96.88	96.88	96.87	96.87	2400	11.3

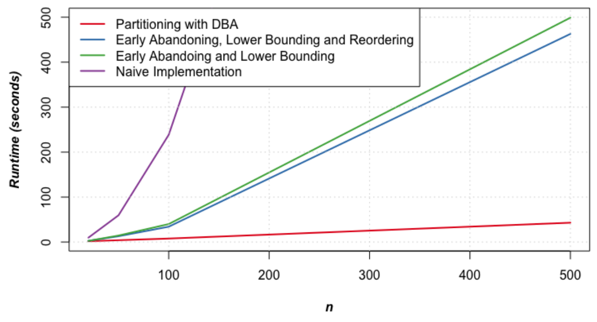

Fig. 7. Runtime of 3 implementations of 1-NN DTW classifier and the partition based method. Partition based is fast, but less accurate (see CV tables). Nearest neighbor classifier is approximately linear when lower bounding is used.

Finally, we comment on how Precision and Recall are computed for this data with three classes. We simply take the average over the three 1 vs all classifiers.

$$P_{avg} = \frac{1}{3}\sum_{k=1}^{3} \frac{M(k,k)}{\sum_{k'=1}^{3} M(k',k)}$$

$$R_{avg} = \frac{1}{3}\sum_{k=1}^{3} \frac{M(k,k)}{\sum_{k'=1}^{3} M(k,k')}$$

and then in the usual way we obtain $F_{1,avg} = \frac{2P_{avg}R_{avg}}{P_{avg}+R_{avg}}$.

A. Validation

Finally, we are able to return to the validation set which we left out since the beginning. We fix $w = 50$. The partition based classifier has accuracy of 60 percent and $F_{1,avg} = 61\%$. The classifier based on nearest neighbor with DTW has an accuracy of 94.4% and $F_{1,avg} = 94.4\%$. Confusion matrices are given below.

Table III: Confusion Matrix (%) for partition based method

		Actual Class		
		1	**2**	**3**
Pred Class	**1**	23	4	4
	2	9	22	16
	3	2	5	15

Table III: Confusion Matrix (%) for 1-NN DTW

		Actual Class		
		1	**2**	**3**
Pred Class	**1**	34	1	0
	2	0	29	3
	3	0	1	31

IV. Discussion and Conclusion

This project has taught me a lot about data mining for time series data. I was honestly taken aback by how difficult it was. Applying DTW was not quite as straightforward as I had hoped/thought it would be. There is a lot of normalization required, and extracting heartbeats was a major challenge (and not even my end goal). I was quite happy to struggle with it though, and feel that I learned a lot along the way. While expert is probably too strong a word, I feel like I have a solid understanding of how to overcome the challenges with DTW and time series mining. Although the algorithms employed were fairly simple, applying them in this scope required a lot of thought.

If I could do this again, the number one thing I would do would be to try to scale it up more and apply this to a larger dataset. I also chose this dataset so that it would meet the size requirement (actually it was still a little shy at 9GB), and I don't think it was actually ideal for a classification task. The dataset is meant more for FECG extraction. I will definitely keep DTW in my toolbox though, and apply it if I get the chance. I also wrote almost all the code from scratch, because I think it's a great way to learn, but I probably wouldn't do it again.

Finally, I wish I had the time to apply more statistical methods to this problem. I wanted to try out regularization methods such as LASSO and the Bayesian Horseshoe, but never really had the time to see if it would work out. The Extremal Depth stuff is also very interesting but needs a better implementation before it is feasible to use on this scale. One of the most interesting things I've learned this semester (and it was reinforced by this project) is the magnitude of difference between statistics and data-mining despite the massive overlap. I love rigorous statistical methods and the quantification of uncertainty that they can provide, but they could definitely learn from and benefit from the people in the data mining community, especially the ability to speed up algorithms in creative ways.

All of the work (code, writeup) in this project is entirely my own. The slides on DTW by Dr. Mueen and Dr. Keogh proved incredibly useful. I tried to write algorithms myself whenever possible, but I have cited all packages that were used.

References

[1] Andreotti F., Behar J., Zaunseder S.,Oster J. and Clifford G D., An Open-Source Framework for Stress-Testing Non-Invasive Foetal ECG Extraction Algorithms. Physiol Meas 5, pp. 627-648, 2016.

[2] Goldberger AL, Amaral LAN, Glass L, Hausdorff JM, Ivanov PCh, Mark RG, Mietus JE, Moody GB, Peng C-K, Stanley HE. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23):e215-e220 [Circulation Electronic Pages; http://circ.ahajournals.org/cgi/content/full/101/23/e215]; 2000 (June 13).

[3] Abdullah Mueen, Eamonn J. Keogh: Extracting Optimal Performance from Dynamic Time Warping. KDD 2016: 2129-2130

[4] Naveen N. Narisetty and Vijayan N. Nair (2016) Extremal Depth for Functional Data and Applications, Journal of the American Statistical Association, 111:516, 1705-1714, DOI: 10.1080/01621459.2015.1110033

[5] R Core Team (2017). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

[6] Thomas Quinn (NA). eek: A Tool to Pre-Process Bulk EKG Data and Detect Physiological Peaks. R package version 0.0.0.9000. http://github.com/tpq/eek

[7] Alexis Sarda-Espinosa (2018). dtwclust: Time Series Clustering Along with Optimizations for the Dynamic Time Warping Distance. R package version 5.5.1. https://CRAN.R-project.org/package=dtwclust

[8] Wang, H. and Song, M. (2011) Ckmeans.1d.dp: optimal k-means clustering in one dimension by dynamic programming. The R Journal 3(2), 29-33.