EMS Motion Data Visual Analytics

Deirdre Scully
Vanderbilt University
2301 Vanderbilt Place
Nashville, TN 37235

Abstract

This project analyzes motion data collected during emergency medical services (EMS) in order to classify which medical procedure occurred during that time. The motion data collected consists of many different types and featurs along with several classes of procedures. The goal of this project is Mixed-Initiative Visual Exploration, to learn information from the collected data through visual and machine learning exploratory analysis.

Introduction

This project analyzes motion data collected during emergency medical services (EMS) in order to classify which medical procedure occurred during that time. The GitHub repository for this project can be found here: `https://github.com/dmscull11/EMS-Prediction`

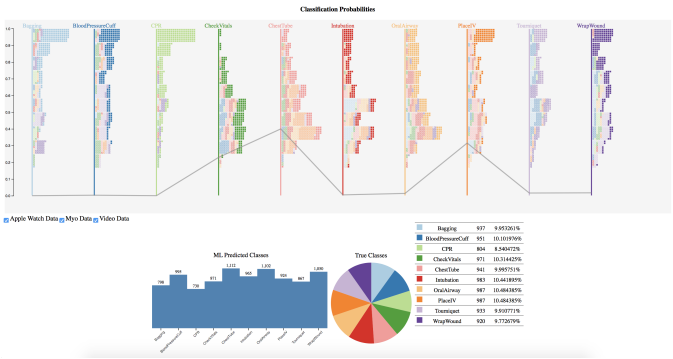


Figure 1: EMS Motion Data Vis

Description

The motivation behind EMS motion data collection is to solve the problem of information loss during patient handoff after an emergency transport, as well as send such information in advanced. The receiving medical facility usually is only given a one paraphrase description of what happened to the patient, such as ”a gunshot wound to the leg”. It would improve patient care if in addition the receiving medical facility was given a comprehensive list of all procedures performed on the patient before arrival, such as ”insert an IV, then CPR for 5 minutes followed by intubation”. This project aims to explore if the EMS motion data is a good classifier of the procedures occurring in real time, as well as which data types and data attributes are necessary and sufficient to do so. In order to answer these questions I explored the data in depth to find trends, similarities, differences, and prediction powers. Visual analytics are be the key component in this data exploration, by implementing useful and interactive data summary and machine learning visualizations of the data and models. The idea is to gain insight into how well and which data types and features classify the various procedures.

Background

The main importance of this project is that it has the potential to help save peoples’ lives in emergency situations. However, the importance of exploring with visual analytics in this project is to determine the best visual encodings for the different combination of data types, features, and machine learning algorithms of this time series motion data. The main difference in this project between other time series data studies is combining and comparing many different types of data into one analysis. Normally in time series data one data item will have many attributes recorded over a period of time. In this data however, one sample is comprised of a few different types of data items that are recorded at different rates per second over a period of time. This means that there is not a direct 1-to-1 correlation within one sample across the types of data items over time. In addition each type of data item has its own distinct set of features, so that direct comparison of the different data types becomes difficult and meaningless. The challenge is then not only how to explore a large amount of data over a series of time points, but also how to compare and analyze across types of data items that are inherently different representations, formats, and with unequal time points. Specifically the Open-Pose spacial output is an interesting new type of data features and application of visual analytics.

Data

All data collected for this project is time series data. Each section of time is labelled with a categorical class. One data type is the video recordings that were collected for most of the data items on somewhere between 0 to 4 camera angles. The different cameras used record at various Hz per second, starting at 24. The video data attributes are the time series image matrices. In addition, the videos are analyzed with the open source Open-Pose limb detection software, which outputs 18 body points by their x and y locations and confidence intervals for each person identified.

Another data type is recorded by apple watches. The apple watches record various data points at 60 Hz per second. The attributes of this data are either the left or right wrist watch, X, Y, and Z acceleration, raw, pitch, and roll.

The last data type is recorded by Myo armbands. These armbands record acceleration data at 60 Hz per second and can detect small muscle movements of the arms (EMGs) which are recorded at 200 Hz per second separately. The attributes of this data are either the left or right arm band, X, Y, and Z acceleration, and 8 different EMGs.

Most of the data is already stored in a PostgreSQL database and csv files, except for the videos which are on AWS in avi and json format. There was great effort needed to quality control the data to make sure everything is labelled correctly as apple watch or Myo, left or right, time stamps include the correct procedure class, separate instances of different patients and procedures are distinguished, the time stamps of each recording device and procedures are following the same wall time, etc. In addition, it takes a long time to pull all of the separate data types into similar formats and imputate the data to be the same lengths and joining over the time points. The data imputation allows to data to be analyzed equally in the machine learning algorithms, but creates sparse data matrices.

Baseline

One interesting visualization approach to time series data uses Temporal Multidimensional Scaling (TMDS), (Jackle et al, 2016). The main usage for this method is to examine time series data that does include many dimensions, or data attributes. It performs dimensionality reduction (DR) on the multidimensions, but still maintains the time series dimension so that the user can explore patterns over time of the large data set. "The x-axis represents the time, and the y-axis represents the MDS similarity value." (Jackle et al, 2016). The time axis will also be continuous instead of discrete, so unequal timestamps will not be an issue. An example of this is shown in Figure 2.

Another approach I considered works similarly with categorical time series data. One algorithmic method von Landesberger et al (2012) implement is clustering the time chunks together based on weather data and attributes. The clusters are then visually encoded as shown in Figure 3. This could apply to my data in an interesting reverse engineering way by clustering the different types of data items into time chunks, and the user could then explore if these clusters of time correspond at all to the already time labelled

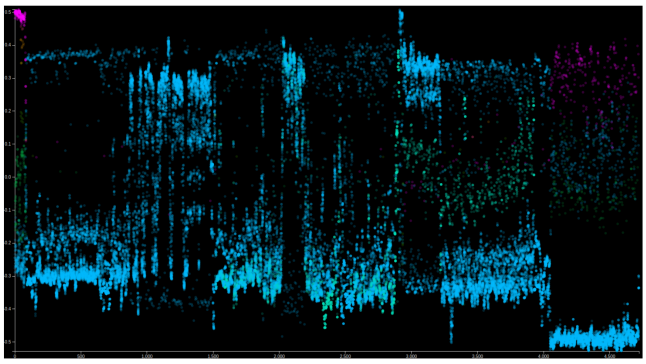
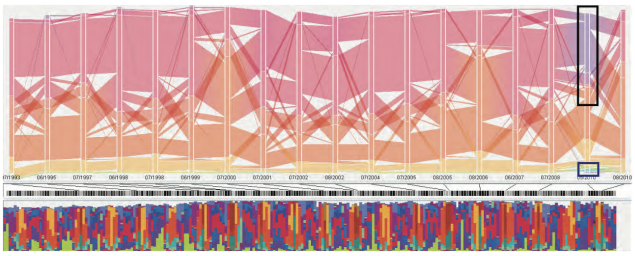
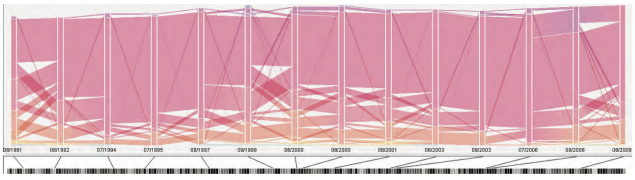


Figure 2: TMDS (Jackle et al, 2016)

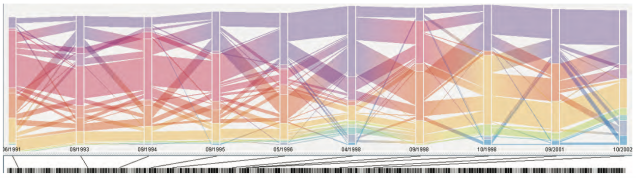
procedures. To do so, some visual encoding of the procedure classes must be included.



(a) Cluster 1: Summer months with predominantly warm weather



(b) Cluster 2: Summer months with stable extremely warm weather



(c) Cluster 3: Spring and autumn with changing weather conditions

Figure 3: Clustering (von Landesberger et al, 2012)

In order to implement either of the ideas mentioned, some changes must be made to work with my data set. The main addition I plan to implement to help visualize my data specifically is a couple filtering methods. The first filter will allow the user to select one or more of the different types of input data items, and follow those specifically throughout the dimensionality reduction and machine learning processing. This selection will still keep the context of the other types of input data items, so the user can easily compare which type of data item inputs are creating what kinds of changes to the models. Another filter, which is really optional labeling, is the addition of a class distinguishing label

in order to follow the data of each procedure class throughout the dimensionality reduction and machine learning processing. This will help the user compare differences and similarities in the time series trends across procedures. These filters when used together can provide many insights into which combination of types of data items are most similar to each other, and predict which procedures that are most dissimilar from the other procedures.

The language used for this project will either be Python or R, and I am already well practiced in both. If Python is implemented D3, Bokeh, or NVD3 (through Django web-app) will likely be the visualization package used. D3 and NVD3 have a larger learning curve, although I am already familiar with NVD3 through Django. The machine learning package used with Python will likely be TensorFlow or scikit-learn, or both for the neural networks and clustering algorithms respectfully. If instead R is implemented for the project the visualizations will be created in a Shiny app. The machine learning algorithms will be used with packages such as nnet for neural networks, cluster for clustering, or other built-in algorithms.

Machine Learning Details

For this analysis and data visualization, the data is classified into one of 10 different medical procedure events. The machine learning algorithm used to classify the data is a random forest of decision trees. Each decision tree classifies the input data as one of the events by continually splitting the training data based on its features and classes. Many decision trees are created and the final class prediction is decided by this set of trees voting, where the mode becomes the predicted class.

Boostrapping was used in this algorithm, so that each time a decision tree is created the input data used is randomly sampled with replacement from the training data set. K-fold cross validation was also used in this algorithm. As the algorithm trains, it is trained on k-1 subsets of the data and the evaluation metrics are evaluated on the last remaining split of the data. This is used to converge the accuracy of the model during training. These algorithm methods allow for the training of the model to have a larger and more diverse set of input data, rather than what is simply given. This is helpful as one problem with this data set is the small sample size for every class, which was as low as 3 samples in certain cases. The classes were also uneven, so in the final models only an even sample of each class was used to train and evaluate the model. In addition, the number of classes was decreased from 25 to 10, by removing some smaller sets of classes and combining some others.

Originally, a convolutional neural network (CNN) was trained to model the data, but due to the small data set and sample size, the algorithm was unsuccessful and only predicted two different classes the majority of the time. Instead, the random forest algorithm was implemented. One main difference between these methods for this particular data is that the CNN was able to treat one sample, all the time points and many features, as an image-like-matrix, learning information from the relation across features and also across time. In contrast, the random forest classifier treats each individual time point and set of features as a completely different input sample. This approach gains the advantage of having more input data, but loses the advantage of learning trends over time. In the learned models and data visualization the probability for each class is actually calculated by the ratio of individual time points voting for the predicted class within one input data set, instead of calculating the voting percentages of the many decision trees.

Visual Encoding Analysis

The final data models visualization dashboard is shown above in Figure 1. It is made of three different graph data encodings and two different tabular data encodings. The main influence for the dashboard is the Squares implementation by Ren et al in 2017, which is also similar to the approach by Kahng et al in 2018 as shown in Figure 5. This visualization encodes how data is classified correctly and incorrectly. As shown in Figure 4, there are different levels of details available for this visual encoding. The most detailed level was implemented because it is nicely complemented by the bar and pie charts that give a general and summary view of the same data.

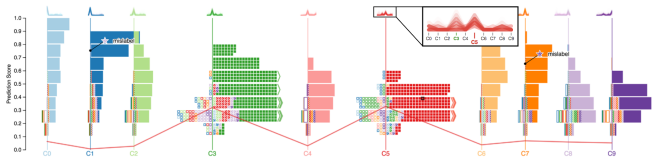


Figure 4: Squares (Ren et al, 2017)

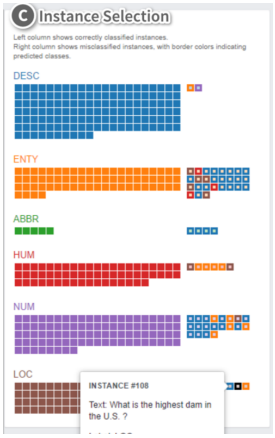


Figure 5: ACTIVIS (Kahng et al, 2018)

The squares implementation encodes several pieces of data, as shown in Figure 6. The main idea is to be able to see which data samples are correctly classified and which ones are not, specifically as which incorrect class instead. The true class of that data sample is encoded by a unique color hue. This color hue is kept the same throughout the entire dashboard for any data that belongs to that true class.

The classes that were predicted by the model are encoded by the x-axis spaces, next to a labeled separating bar of that color hue. Correctly classified samples are encoded with solid squares and incorrectly classified samples are encoded as striped squares. The probability of the predicted class for each sample, calculated by the voting time points, is encoded by the y-axis scale 0 to 1. Hovering over a specific sample the corresponding model probability values for every class of that sample are shown with a linear coord across the separate sets of class squares.

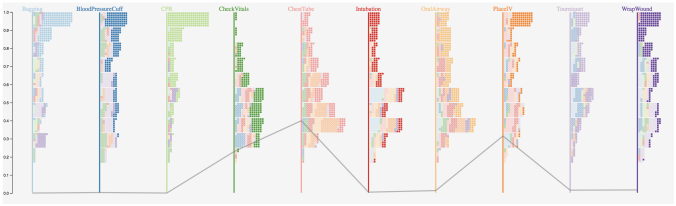
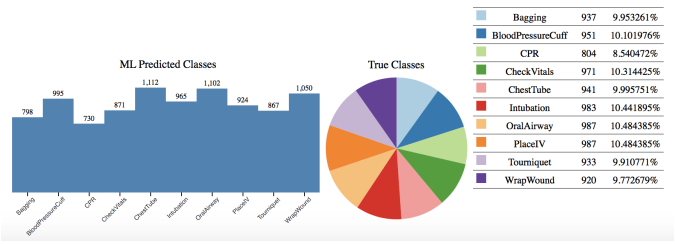


Figure 6: Dashboard Squares

The lower visualizations are directly linked with each other, as shown in Figure 7. The pie chart shows the proportion of samples belonging to that true class, encoded by size for the number of occurrences and by color hue for the true class. The table to the right lists these true classes, their actual frequencies, and their percentage frequencies. This table is also encoded with true class color hue, and separates the classes across the y-axis with separating lines. The class name, frequency, and percentage frequency is encoded across the x-axis. The bar chart to the left, represents for which class the model predicts those samples. To summarize the squares implementation the pie chart shows the total true classes, and the bar chart shows the total incorrectly classified samples for each true class. The bar chart is therefore encoded with color hue of the true class that sample belongs to, to keep consistency across the dashboard. The incorrect classes are encoded along the x-axis and the frequency of each is encoded by the y-axis of bar height, with the frequency label shown in addition. Hovering over one of the bar classes will update the pie chart to depict which classes are actually the true classes for those predicted samples. In reverse, hovering over the pie chart will update the bar chart to depict how those true samples were actually classified by the model.



Bagging	937	9.953261%
BloodPressureCuff	951	10.101976%
CPR	804	8.540472%
CheckVitals	971	10.314425%
ChestTube	941	9.995751%
Intubation	983	10.441895%
OralAirway	987	10.484385%
PlaceIV	987	10.484385%
Tourniquet	933	9.910771%
WrapWound	920	9.772679%

Figure 7: Dashboard Bar and Pie

Across the entire dashboard the true class of the data is encoded by a unique color hue. When hovering over specific data with the mouse, such as individual squares, bars, or pie slices, the color intensity is changed to make that section more noticeable. The squares implementation is given a slightly tinted background to make the two sets of data encodings appear more distinct. These two views are joined by the checkboxes interaction, as shown in Figure 8. These checkboxes encoded the three different types of input data that have their own unique set of features. The model was separately run on each of these sets of input features and every combination of the two or three of them. Enabling or disabling each of the checkboxes updates the top and bottom views of the data model output with the results from using that exact combination of data input features.

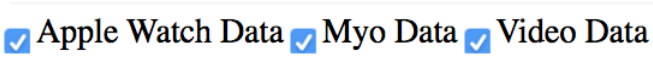


Figure 8: Dashboard Checkboxes

This visualization encoding of the model data allows the user to evaluate how different types of features are classifying the data incorrectly. This gleams information such as groups of similar classes within that set of features. The squares implementation, as shown in Figure 6, gives the user interaction to see granular individual samples, how they are classified or mis-classified, and the probabilities of how close they might have been predicted as other potential classes. The lower view, as shown in Figure 7, gives the user a more readable and overview summary of what is encoded in squares. The interaction between the two is created by the user updating different sets of features for both visualizations simultaneously.

Baseline Comparison

One of the biggest differences from the proposed baseline to the final dashboard is the usage of time. After spend a while pre-processing, imputating and trying out visualizations it became clear this data isn't a great example of time series data. Since each procedure is a different length of time, after the data imputation the features over time can be very spare for certain classes of procedures. Varying from 10 seconds to 10 minutes long, comparing features at specific time points has no real measured value for this data set.

Another idea that was added and then discarded through the dashboard iterations was the use of the input features hierarchy breakdown, visually encoded as a tree. This visual encoding was interesting and helpful for understanding the specific features in the set of input data, but was determined to add minimal value to the overall dashboard. It also had no interaction with the squares implementation, only inadvertently through the related information given in the interactive checkboxes for updating those sets of input features.

The main similarity from the baseline is the squares implementation for the purpose of viewing how samples are classified and seeing these different class groupings for each set of input features. One idea since the baseline was to implement two squares, one being static with only the model from using all of the data and the other changing with the

user input features checkbox selections. Even with the least granular detailed squares implementation this idea left no interaction between the two sets of plots, just contextual comparison. The current visualization gives interaction with simultaneous data updates for both sets of plots. Without this static view the user does lose the contextual information of comparing to the all features included plot, however the goal was more to learn groupings of classes within a set of features. The current visualization tool gives the user that ability to see detailed mis-classifications, but also have a big picture summary of the same data as a "zoomed in and out" contrasting views.

Data and ML Analysis

The data visualization dashboard is definitely successful in allowing the user to see classification and mis-classification trends across a set of features. The squares implementation allows the user to see how high or low the probability was in any mis-classification. Some smaller groups of mis-classified samples can be difficult to determine, so the overview method gives the user the ability to see the overall distribution of how every class of samples were predicted.

Trends in the predicted classes can easily be seen by the user for some of the model output, given each set of input features. For example, considering just the Apple Watch motion data CPR has the highest accuracy. This makes sense given its distinct and repetitive motion compared to all the other procedures. Wrap wound, Tourniquet, and Blood Pressure Cuff have a high overlapping of mis-classifications which makes sense since they all have an aspect of circular motion occurring during the procedure. Considering just the Myo EMG muscle data Chest Tube, Intubation, Oral Airway, and Place IV all have a high overlapping of mis-classifications which makes sense since they all require a similar amount of steady muscle tension of the medic's hands and arms. Considering just the Video location data Bagging, Intubation, and Oral Airway all have a high overlapping of mis-classifications which makes sense since they all take place around the patient's head.

The machine learning algorithm was generally not very successful with rather low accuracy rates. The lowest accuracy was for the Myo EMG set of features and the highest was for the Apple Watch Accelerometer set of features. The models do offer important information such as groups of similar and dissimilar classes based on the different input features, some examples noted above. This is the most important take away from the model visualization dashboard because if one was simply using accuracy the user would only see poor overall performance and learn nothing from the model. With the data visualization tool, despite poor prediction performance, we are still learning grouping information from the data.

References

T. von Landesberger, S. Bremm, N. Andrienko, G. Andrienko and M. Tekuov. 2012. Visual analytics methods for categoric spatio-temporal data. *IEEE Conference on Visual Analytics Science and Technology (VAST), Seattle, WA, 2012*. pp. 183-192.

D. Jackle, F. Fischer, T. Schreck and D. A. Keim. 2016. Temporal MDS Plots for Analysis of Multivariate Data. *IEEE Transactions on Visualization and Computer Graphics.* vol. 22, no. 1. pp. 141-150.

Minsuk Kahng, Pierre Y. Andrews, Aditya Kalro, and Duen Horng (Polo) Chau. 2018. ACTIVIS: Visual Exploration of Industry-Scale Deep Neural Network Models. *IEEE Transactions on Visualization and Computer Graphics.* vol. 24, no. 1.

Donghao Ren, Saleema Amershi, Bongshin Lee, Jina Suh, and Jason D. Williams. 2017. Squares: Supporting Interactive Performance Analysis for Multiclass Classifiers. *IEEE Transactions on Visualization and Computer Graphics.* vol. 23, no. 1.