Smartphone-Based Intelligent System: Training AI to Track Sobriety Using Smartphone Motion Sensors

Undergraduate Research Thesis

Presented in partial fulfillment of the requirements for graduation with *honors research distinction* in Computer and Information Science in the undergraduate colleges of The Ohio State University

By
Jackson Killian

The Ohio State University
April 2018

Project Advisors: Professor Kevin Passino, Department of Electrical and Computer Engineering; Associate Professor Arnab Nandi, Department of Computer Science and Engineering

Abstract

Excessive alcohol consumption is an avoidable health risk, yet it causes a significant percentage of yearly deaths and injuries on college campuses. Recent work showed that weekly mobile-based interventions can effectively reduce alcohol consumption in students. However, few studies investigate delivering mobile interventions in real-time during drinking events where interventions could reduce risks like drunk driving, alcohol poisoning, and violence. Such studies require measuring real-time intoxication levels outside of a lab setting at scale. Some technologies exist for this purpose but are impractical or expensive. To address these shortcomings, we built an intelligent system capable of passively tracking smartphone accelerometer data to identify heavy drinking events in real time. We collected smartphone accelerometer readings and transdermal alcohol content (TAC) readings from 20 subjects participating in an alcohol consumption field study. The TAC readings served as the ground-truth when training the system to make classifications. The TAC sensors and smartphone accelerometers both provided noisy readings which were cleaned with the MATLAB signal processing toolbox. We then segmented the data into 10 second windows and extracted features known to change when humans lose control of their center-of-mass (i.e. become intoxicated). Additionally, we experimented with some feature extraction methods from sound recognition tasks and show that they provide a significant improvement in this task (up to 8% absolute accuracy gain in our case.) Finally, we built and trained several classifiers to call each window as a “sober walk” or “intoxicated walk”, the best of which achieved a test accuracy of 75.04%. This result has promising implications for making classifications on noisy accelerometer data in this space and also offers multiple avenues for improvement. We plan to use our classifiers to build a mobile sobriety tracking application that ultimately will serve as a free, reliable, and widely adoptable application that tracks intoxication in real-time, enabling development of effective real-time mobile-based interventions which can later be delivered via the application to reduce unnecessary alcohol-related injury and death. The results and application will also benefit future studies as new sensor-bearing technologies become widely adopted.

Introduction

Excessive alcohol consumption is an avoidable health risk, yet a significant percentage of deaths globally can be attributed to the drug [1]. On college campuses, where binge drinking is common, alcohol-related risk is especially dramatic. Recent work has shown that weekly mobile-based interventions can be effective in reducing alcohol consumption in students [2]. However, few studies characterize the efficacy of delivering mobile interventions in real-time during heavy drinking events; though such interventions could theoretically reduce event-level risks such as drunk driving, alcohol poisoning, and violence by ending drinking episodes before these outcomes. Such studies will not be desirable until a method exists for measuring real-time intoxication levels outside of a study setting at large scale. Some technologies exist for this purpose, but those currently available are impractical (user must answer surveys), expensive (smart-watches), or both (ankle bracelets). To address these shortcomings, we propose a smartphone-based system to passively track a user’s level of intoxication to support the delivery of mobile-based interventions in real-time during heavy drinking events.

Prior Work

Applying machine learning to classify levels of intoxication has recently gained popularity. In 2012, Kao et al. [3] conducted a small lab study with three participants to determine features of a person’s gait that change significantly with levels of intoxication, namely the length and speed of one’s stride. In 2015, Arnold et al. [4] trained an intelligent system on accelerometer data from a smartphone to classify with up to 70% accuracy the approximate number of drinks that participants consumed. However, a small sample size (N=6) and the use of self-reports to establish intoxication levels limited the reliability of their model. In 2017, Bae et al. [5] built on this idea by conducting a field study (N=38) to gather a multitude of smartphone sensor data such as keystroke speed, sent/received calls, location and more. Their model correctly classified 96.6% of self-reported drinking episodes, allowing for theoretical intervention within the half-hour. This level of accuracy is promising. However, the intoxication level of participants was again established using self-reports which the authors note is susceptible to biased, inaccurate, or missing data. Additionally, the accuracy of their model relied on constantly sensing and storing data for up to 3 days at a time, even outside of drinking events. Such prolonged activity can drain smartphone resources such as battery life, potentially forcing a user to stop using the application in a real-world setting. Additionally, collecting and storing so much personal information poses privacy concerns that might further deter users. Later in 2017, Gharani et al. [6] carried out a field study (N=10) which built fairly reliable linear regressors ($r \geq 0.9$) based solely on gait features measured from an accelerometer. However, their data also relied on biased self-reports to establish ground truth levels of intoxication.

Our Contributions

In this study we make several key advances and contributions in the area of using gait to classify sobriety as follows:

1. *High-Quality dataset:* To the best of our knowledge, we carried out the first medium-scale field study (20 participants) which used sensors to establish ground-truth levels of intoxication rather than by self-reports. Thus our findings are applicable to real-world drinking scenarios and our classifications are free from user bias.
2. *Discovered Powerful New Features:* We experimented with a new set of features previously unused in the literature for gait analysis for classifying sobriety. We demonstrate that these features improve classification accuracy by as much as 8%.
3. *Trained and Compared Several Different Classifiers:* We train a convolutional neural network, long short-term memory recurrent network, shallow neural network, random forest, and support vector machine to make classifications of sober (TAC <= 0.08) vs. intoxicated (TAC > 0.08) and compare their performance by class population. The support vector machine performed the best overall (75.04% accuracy) while the random

forest provided the most generally reliable results per class (77.43% accuracy for sober data vs 68.45% accuracy for intoxicated data.)

Methods

Data Gathering

Our field study consisted of 20 college students who were all participating in a yearly informal "bar crawl" event held independently by bars on our college campus. Through the course of the ~18 hour event, students voluntarily engaged in heavy drinking activities. The 20 participants we recruited were offered moderate financial compensation to share mobile data, periodically answer questionnaires and wear a sensor for measuring transdermal alcohol content (TAC) throughout the event. In this thesis, we focus solely on accelerometer data and TAC data collected from participants. To gather accelerometer data, we developed an application (on iPhone and Android) to measure X, Y, and Z axis readings (sampled at 40Hz) which were periodically sent to and stored on an InfluxDB server [7]. The TAC data (sampled every 30 minutes) was collected using a SCRAM ankle bracelet sensor [8]. (IRB approval number: 2016B0092) All data was fully anonymized after collection.

Data Cleaning

Our custom application for collecting accelerometer data failed to properly install on 1 participant's mobile device. Of the remaining 19 participants, the company which provided our final TAC readings (SCRAM) reported that 6 of the sensors produced data that was indicative of a malfunction (power failure, total interference from clothing, nonsense readings, etc.) The rest of this thesis focuses on the data from the remaining 13 participants.

The TAC sensors (prone to splashes of alcohol or loss of skin contact) and smartphone accelerometers (relatively inexpensive) both produced relatively noisy readings. To smooth both sets of data, we employed MATLAB's signal processing toolbox which has several pre-built filters capable of removing noise above a given frequency, *fstop*. We found that a Chebyshev Type II filter, known for its steep roll-off, worked well empirically for smoothing both sets of data [9]. Note that the higher the order of the filter, the steeper the roll-off. For the TAC data, we used a 10th order Chebyshev Type II filter and *fstop* = 1e-4Hz. The order and *fstop* were again determined empirically. We applied the filter in the forward and reverse direction in order to maintain phase, so that the cleaned TAC could still reliably be matched to corresponding accelerometer readings through their timestamps. Note that this is important since the sampling frequency of the TAC data is over a much larger time scale than that of the motion data, so small phases changes could result in the TAC readings being shifted by several minutes. Then, to account for the time it takes for alcohol to exit the bloodstream and evaporate through the skin, we subtracted 45 minutes from each TAC reading to obtain readings more indicative of real-time blood-alcohol content (BAC) [10]. For each axis of accelerometer data we found empirically that a 15th order Chebyshev type II filter worked well and we set *fstop* = 2.7Hz since the average human walking frequency is about 2Hz [11] which we also confirmed empirically.

Segmentation

To make the data more manageable for input to a machine learning classifier, we segmented each stream of ~18 hour time-series accelerometer data into windows in a range of a few seconds to a few minutes. We carried out this segmentation in two steps. Over the course of the bar-crawl event, participants' mobile devices experienced occasional periods in which they lost internet connection or battery power. These cases resulted in either zero-readings or a lack of data collection altogether for the duration of the outage. Hence, for our first step, we broke each participant's stream of data into segments separated by at least two minutes of zero-data or a lack of readings, which we will call data-sessions.

For the second segmentation step, we tried two approaches. For the first, we set out to isolate segments of accelerometer data that indicated that a participant was walking. To accomplish this, we used a sliding window over each data-session (width of 4 seconds) in which we analyzed the frequency content of the data, keeping windows which were rich in frequency content near 2Hz. Consecutively accepted windows were concatenated up to a maximum length of 60 seconds, resulting in windows of data between 4 and 60 seconds. For our second approach, we blindly segmented each data-session into windows of length *x*, where *x* ranged from 4 seconds to 2 minutes, and each window was required to have at least 90% data density (i.e. a 4 second window at a sampling rate of 40Hz must have at least 144 data points or ~3.6 seconds of data.)

Feature Extraction

In preparing time-series data for classification by machine learning algorithms, it is common to extract features from both the time domain and frequency domain. In both cases, it is also common practice for each calculated feature to use a two-tiered windowed approach to characterize the data as it changes with time. For example, for a given measure/metric of some 4 second segment, one might further segment the window down to one second segments, calculate the measure/metric for each small segment, then use the mean, variance, and median of the measure/metric over the 4 smaller segments to characterize how it changes over time, effectively creating many features to feed to a classifier.

Short Term Window Features

Kao et al. [3] found that gait stretch (length of step) and step time (time to complete a step) changed as participants became intoxicated. Arnold et al. [4] also explored as features gait cadence, signal skewness, and signal kurtosis in the domain, as well as average power and ratio of spectral peaks in the frequency domain. We calculate each of these as well as several others as the basic features to input to our classifiers. The full list is below in **Table 1.**

Feature(s)	**Definition**
Mean [12]	Average of raw signal
Standard Deviation [12]	Standard deviation of raw signal
Median [12]	Median of raw signal
Zero Crossing Rate [12]	Number of times signal changed signs
Max, Min [4 total] [48]	Max/Min of raw and absolute signal
Spectral Entropy [24]	Entropy of energy in both the frequency and time domain
Spectral Centroid [12]	Weighted mean of frequencies
Spectral Spread [12]	Measure of variance about the centroid
Spectral Flux [12]	Measure of speed of change between two consecutive FFTs
Spectral Roll-off [12]	Frequency under which 90% of energy is contained
Max Frequency [6]	Max frequency from FFT
Gait Stretch [6]	Difference between max and min of one stride
Number of Steps [6]	Total steps taken during a window
Step Time [6]	Average time between two steps
Cadence [6]	Total steps over total time
Skewness [6]	Measure of asymmetry of time-series signal
Kurtosis [6]	Heaviness of tail/Fourth moment of time-series signal
Average Power [6]	Average power over a Welch's power spectrum distribution
Spectral Peak Ratio [6]	Ratio of largest peak to second-largest peak

Table 1: Features calculated per short-term window. Each short term feature becomes the basis for 6 summarizing statistics downstream. Each of the features described in the first 10 rows are calculated per window, then used again to find the difference between the current and previous window.

MFCC Covariance Features

To add to the list of features commonly extracted in studies such as this, we drew from a technique sometimes used in speech recognition tasks. The technique first uses a method which summarizes the energies of a signal in the frequency domain into 13 bins, known as Mel Frequency Cepstral Coefficients (MFCC). The technique requires one to calculate the MFCCs for a given small windows of a segment (i.e. 1 second windows) then calculate the covariance matrix via the output matrix dotted with its own transpose. For example, if a 4 second window was split into 1 second segments, one would calculate 13 MFCCs over 4 segments resulting in a matrix M of dimension 13x4. The covariance matrix would be calculated as MM^T. In this way, the covariance matrix of the MFCCs captures the change in frequency content of the signal over time. In our experiments, we flatten the covariance matrix and keep only the entries above the diagonal since the resulting 13x13 matrix is symmetric. We calculate these 91 coefficients for each axis of accelerometer data with itself as well as for each axis with each other (i.e. X by X, X by Y, X by Z, etc.)

Root Mean Square of Signal
Finally, the root mean square (RMS) of lateral accelerations was shown to be significantly different in a study of transfemoral amputees between control and amputee populations [12]. We posited that this quantified a lack of control of one's center of mass which would also be present in our intoxicated subjects. Thus for each window we calculated the RMS of the signal over each axis.

All of the above produced 1215 features per window of data. Below we discuss a method for determining feature importance to reduce the size of our input vectors for performance reasons.

Classifiers
No single machine learning approach stands alone as a gold standard for any given classification task. Thus, we built and trained five different classifiers to run on our data. All results below were calculated using at least 3-fold cross validation to ensure that none of the given machines were over-fit to any cut of training data.

Multilayer Perceptron Network
A multilayer perceptron (MLP) network is a basic type of feedforward artificial neural network. We chose to use this machine due to its versatility and common use in classification tasks. Our MLP network consisted of three layers: the input layer, one hidden layer of size N consisting of sigmoid activation functions, and two output neurons to which a softmax was applied. Using learning rate η, we trained using gradient descent to reduce the cost function calculated as the average of the softmax cross-entropy between predictions and actual labels. Cross validation established N=256 and $\eta = 0.001$. The network was built using TensorFlow [13].

Long Short-Term Memory Network
A long short-term memory (LSTM) network is a type of recurrent network capable of maintaining a state or "memory" over a period of time. We chose to build an LSTM network because of its common use in time-series prediction tasks. Our intuition was that, given the gradual nature of changes in intoxication, the LSTM may be efficient at predicting sobriety as its previous predictions inform future ones – just as a bartender would watch a customer throughout an evening judging their sobriety gradually based on many snapshots of their behavior. We used Keras [14] to build the LSTM network with one dense hidden sigmoid activation layer of size 256, trained using a binary cross-entropy loss function, Adam optimizer and batch size of 32.

Support Vector Machine
Support vector machines (SVMs) are also very versatile and common in classification tasks. They differ from neural networks in that, rather than stochastically training to find boundaries between classes, they optimize to find exactly the boundary that maximally separates classes.

Additionally, SVMs may provide non-linear decision boundaries by using basis functions that non-linearly project features into a higher dimension before calculating a decision boundary. In the event that all classes are not clearly separable (as is often true), it requires a significant amount of parameter tuning to tradeoff between overfitting and misclassification. We built an SVM with a radial basis function:

$$e^{-\gamma * |\boldsymbol{u}-\boldsymbol{v}|\hat{}2}$$

where γ is a free parameter and **u** and **v** are feature vectors. Note that we also included a cost parameter C in our loss function:

$$min\ |\boldsymbol{w}|\hat{}2 + C\sum\boldsymbol{\xi}$$

where **w** is the weight vector and $\boldsymbol{\xi}$ is the set of slack variables. The larger C is, the more the slack variables are penalized in our loss function, ultimately leading to less misclassification but more chance of overfitting. C and γ were selected via cross-validation as 1 and 4 respectively. We built the SVM using a python wrapper of LIBSVM [15].

Convolutional Neural Network

Convolutional neural networks (CNNs) are a type of neural network that consist of many hidden layers – otherwise known as "deep networks." CNNs are a unique type of deep network that do not require one to extract features before passing it through the network; rather, the network is structured to develop and determine important features internally by consuming only the raw data as input and undergoing supervised training. CNNs accomplish this by using alternating sequences of convolutional layers, which apply a function over a given window of raw data to compute a feature, and max-pooling layers, which simply take the maximum value over some window of data as the feature. These alternating layers are conventionally followed by some number of flat, fully connected layers, eventually terminated by a softmax over a final output layer equal in size to the number of classes. We decided to build a CNN due to its recent popularity in time-series based classification tasks, and due to the fact that, to the best of our knowledge, no study has yet attempted to apply it to this task.

We built a CNN with the following architecture: (8x8x1) → CONV (8x8x32) → POOL (4x4x32) → CONV (4x4x64) → POOL (2x2x64) → FC (256x64) → FC (64x2) → OUT. For input, we transformed the raw time-series data to the frequency domain using a Fast Fourier Transform, then cut it to the first 64 bins (low frequencies), since they contained the vast majority of frequency content. We trained using a dropout rate of 0.5, cross-entropy loss, Adam Optimizer, and used cross validation to determine a learning rate of 1e-5. We built the CNN again using TensorFlow [13] and using open source code as a template [16].

Random Forest

We constructed a random forest classifier as our final machine. Random forests are widely applicable, easy and quick to train and require almost no parameter tuning. They work by blindly and randomly constructing multitudes of decision trees, then taking the mode of the predictions of the underlying trees. The motivation behind this idea is that, although many trees will give spurious predictions, if enough trees are generated then the spurious predictions will all cancel each other out while the few trees making good predictions will be the most common and rise to the top through the mode. This method of generating many random trees also naturally defends against overfitting. Interestingly, random forests have tended to perform very well, if not the best, for previous studies in this area [4, 5]. We built the random forest using Python's Scikit-Learn [17].

Feature Importance and Dimensionality Reduction

As mentioned above, we generate 1215 features per window of accelerometer data. However, inevitably, not all of those features were essential for differentiating between classes, and reducing the size of input vectors can speed up training. We left the fraction-of-most-important-features-to-keep, λ, as a parameter to tune via cross validation. To determine the importance of each feature we trained Scikit-Learn's Extra Trees Classifier on randomized subsets of the data. The trained classifier in turn reported the importance of each feature calculated as the "normalized reduction of the criterion brought by that feature" [17].

Parameter Tuning

There were three parameters that needed tuning in this study regardless of the underlying machine, namely the cutoff point for sober vs. intoxicated, the fraction of important features to keep (λ) and the length of the window. Due to time restrictions, these were calculated using cross-validation on only the MLP network.

Our intuition suggested that the differences between sober and intoxicated would become more pronounced as the level of intoxication rose. However, **Figure 1** shows that this was not generally true until dangerously high levels of intoxication – and even then performance gains were minimal and likely due to a lack of data. Thus we kept the cutoff TAC at 0.08, the legal limit for driving while intoxicated in the United States.

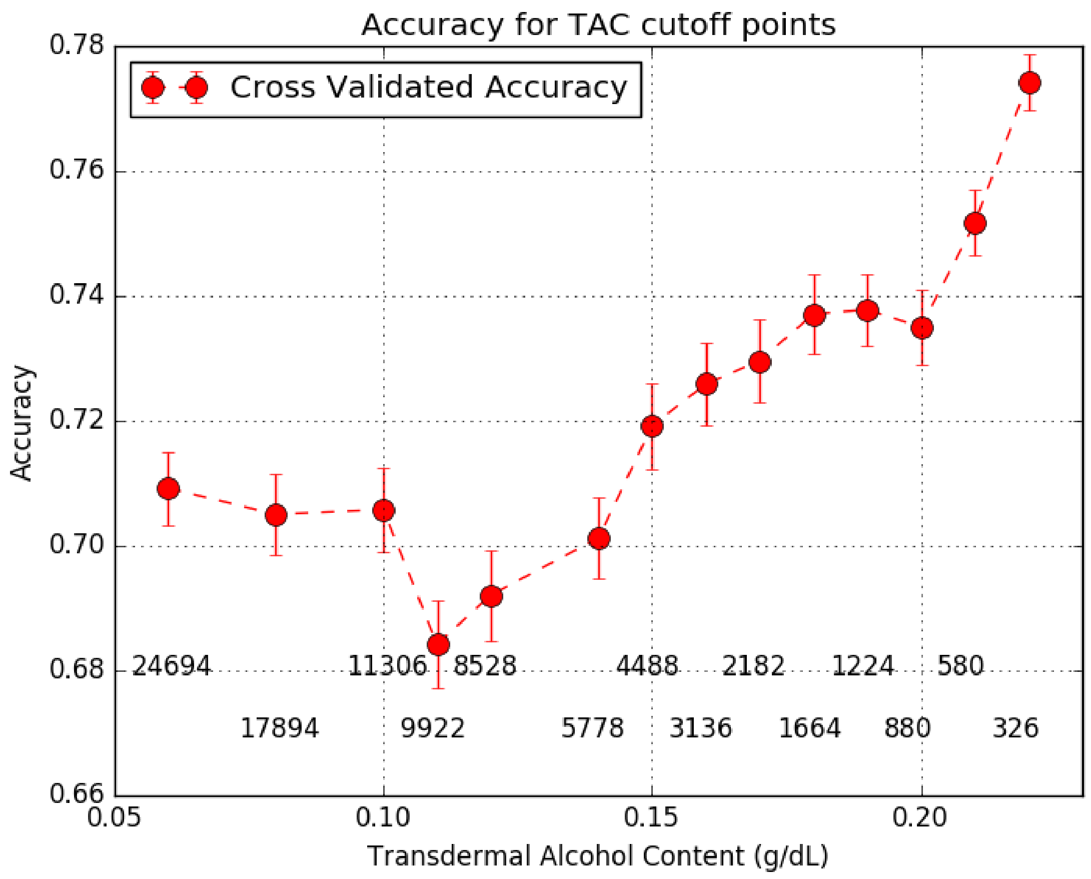

Figure 1: Classification accuracy vs. the TAC cutoff point for sober vs. intoxicated. Some classification accuracy gains were made as the cutoff point was increased, but this was likely a symptom of a lack of data (the number below each point denotes the number of data points in our study with TAC above that threshold) – and gains were minimal.

Below, **Figure 2** shows the performance of our classifier vs. the number of most important features kept. Small gains in accuracy are made until about 0.2, after which no additional improvements can be seen and training becomes cumbersome. Therefore we set λ = 0.2.

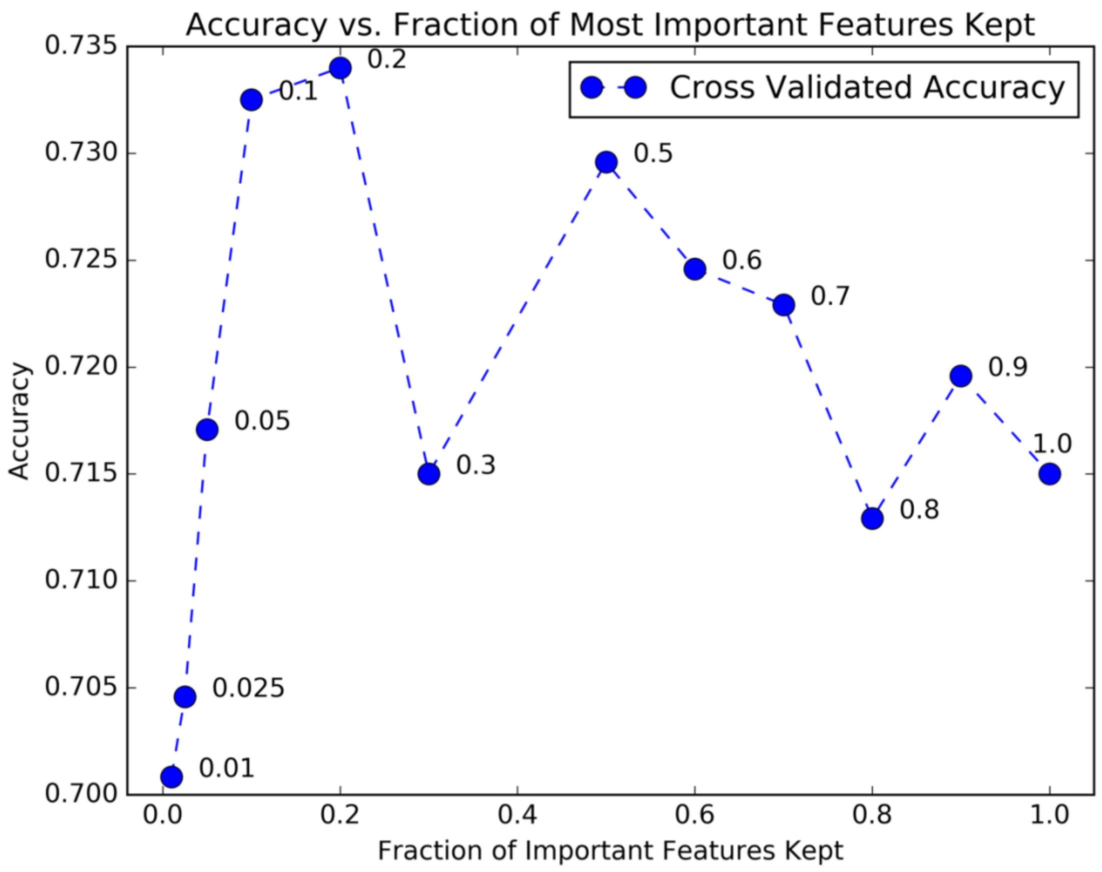

Figure 2: Cross validation run to determine what fraction of the most important features to keep. λ = 0.2 was the clear best choice.

Finally, we posited first that our segmentation method to extract only windows of walking data would provide the best downstream results by reducing noise and narrowing the scope of our classifier. However, using this method we obtained a maximal classification accuracy of 65% (10% less than our best method.) Thus we decided to move forward with blindly cut windows of non-zero accelerometer data. In general with this approach we guessed that an ever-increasing window length would add more information to be described by our feature extraction method giving us better results downstream. However, **Figure 3** demonstrates that the opposite was generally true. We obtained the best performance for our network when using the shortest tested window of 10 seconds. Fortunately, this is desirable for real world applications, since the shorter the window length, the less resources that are expended and more likely one is to be able to gather the necessary data to perform a classification.

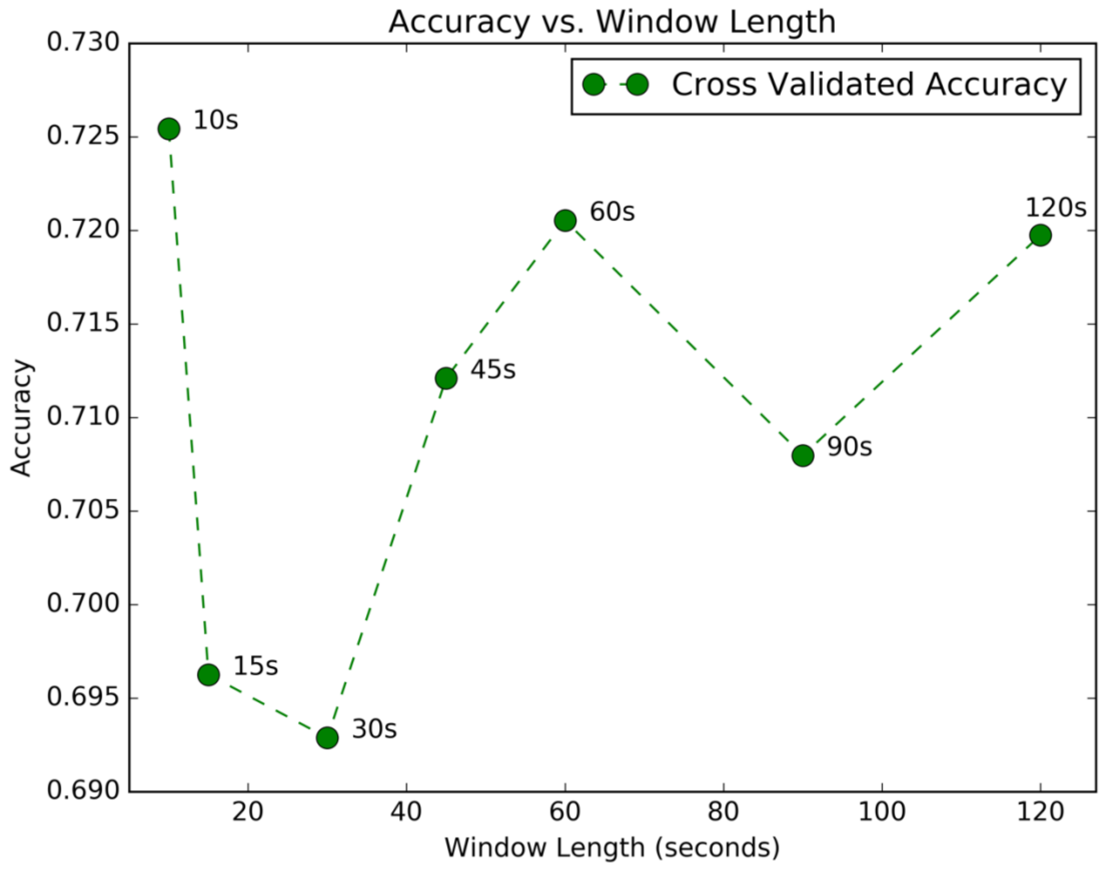

Figure 3: Accuracy for given window lengths. The shortest window (10 seconds) yielded the best classification accuracy.

Results

Using a window length of 10 seconds and $\lambda = 0.2$, we had 26,087 rows of data each with 243 features. Again, each of the results shown here (**Table 2**) are the average from 3-fold cross validation, training on the three non-overlapping splits of 17,391 training samples and 8696 test samples. For the top three classifiers, we also include the accuracy-per-class, precision, and recall.

Classifier	Accuracy	Sober Accuracy	Intoxicated Accuracy	Precision	Recall
CNN	68.20%	-	-	-	-
LSTM	70.15%	-	-	-	-
MLP	72.5%	**81.62%**	53.27%	60.09%	53.31%
Random Forest	74.20%	77.43%	**68.45%**	61.11%	**67.93%**
SVM	**75.04%**	81.54%	61.49%	**63.50%**	61.49%

Table 2: Final classification accuracies, precision, and recall per machine. Bolded results represent the best performance in each column/category. The MLP network achieved the highest accuracy in classifying sober data, but at the expense of both precision and recall – it guesses that most data is sober. The Random Forest and SVM are more robust – notably the Random Forest is clearly the best at identifying intoxicated data.

We also wanted to investigate and quantify the level of classification accuracy that we added with the novel application of MFCC covariance matrices as features. Hence, we retrained and tested the MLP, Random Forest and SVM on the data without the MFCC covariance features, the results of which can be seen in **Table 3**.

Classifier	Accuracy w/ MFCC	Accuracy w/o MFCC	Difference
MLP	73.49%	65.81%	7.68%
Random Forest	74.20%	67.63%	6.57%
SVM	75.04%	66.97%	**8.07%**

Table 3: Classification power added by MFCC covariance matrix entries as features.

Note that since we did significantly change the number of features, before retraining it was necessary for us to retune the parameters for each classifier. Cross-validation showed that $\lambda = 0.2$ was still the optimal fraction of important features to keep for all machines and that N=256 neurons in the hidden layer was still optimal for the MLP network. However, the new optimal parameters for the SVM were C = 8 and $\gamma = 4$.

Finally, since the Random Forest arguably performed the best in general across all categories, we further investigated its robustness by exploring the variance of its errant calls. **Figure 4** shows a box plot of the absolute difference between the TAC of a data point and the 0.08 cutoff level in two cases: when the actual data was sober data but we called it intoxicated (False Positive) and when the actual data was intoxicated data but we called it sober (False Negative).

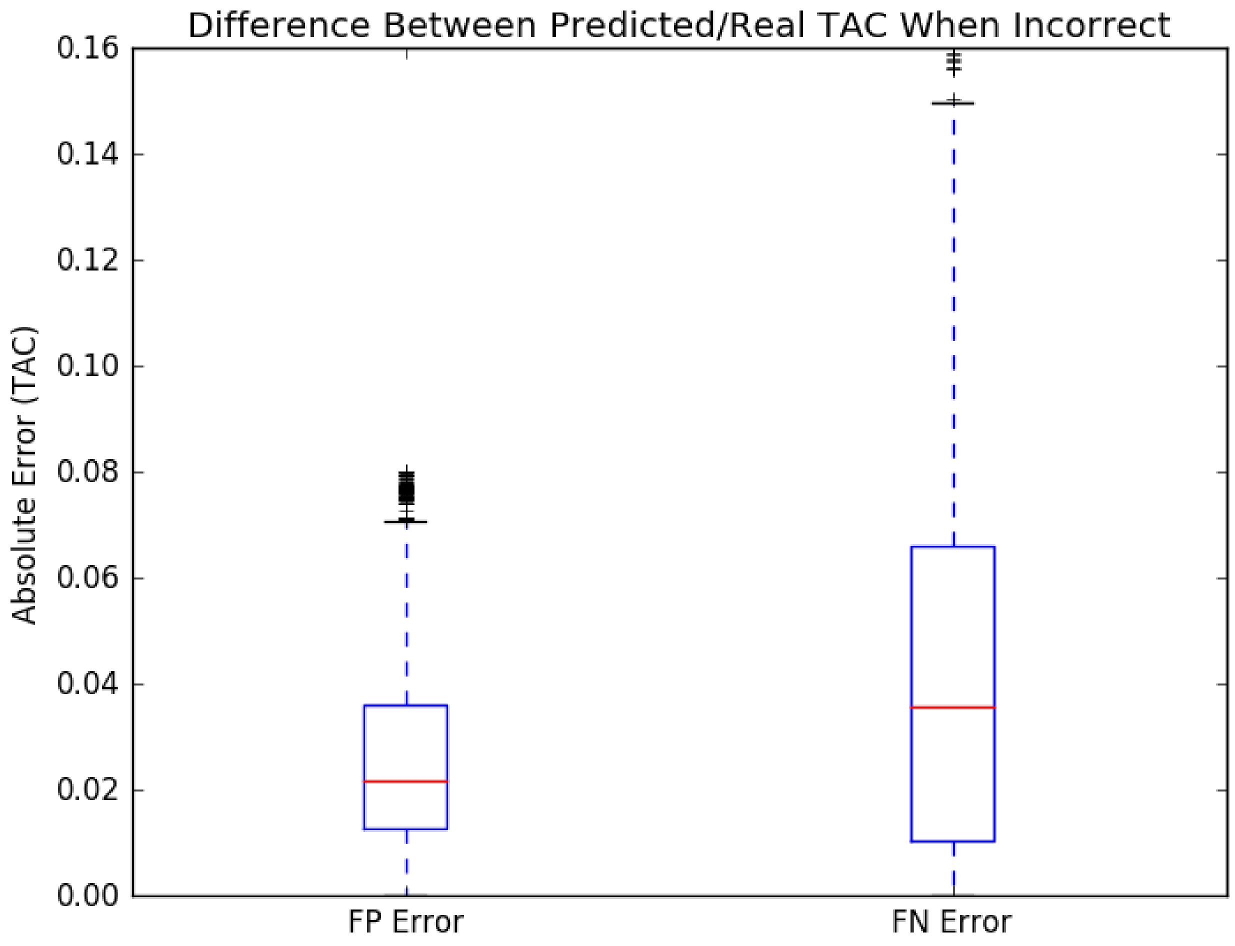

Figure 4: A boxplot describing the difference between the Random Forrest's call (above or below 0.08) and actual TAC in cases where the call was incorrect. The left shows the distribution for false positive cases and the right for false negative cases. The red line denotes the median.

Discussion

To the best of our knowledge, we have achieved the best-to-date binary classification accuracy on an accelerometer/intoxication task, reaching 75.04%. We were able to achieve this result for two main reasons. First, we accessed a participant cohort that was sufficiently large and that generated data for a time-span long enough to allow us to collect more than 25,000 observations, whereas the performance of classifiers of some comparable studies have had far less data. Gathering so much data allowed us to train classifiers prepared for many different input cases without suffering from overfitting. Further, we discovered that calculating MFCC covariance matrices as features drastically improved classification accuracy for all of our machines – achieving as much as an 8% absolute gain.

Interestingly, although the SVM had the best overall classification accuracy, it did not perform exceptionally well when classifying intoxicated inputs – in fact, it was only able to achieve about 11% better than a random guess. However, though the random forest had 1% poorer absolute classification accuracy, it was far better at correctly classifying intoxicated inputs than other machines. Notably, the neural network was the best at classifying sober samples, but almost completely unable to generalize its training to identify intoxicated samples.

Further, though we posited that the RMS of the signal may boost classification accuracy, further inspection showed that it rarely made the cut of the top 20% of most important features, if at all. Far more classification accuracy was gained using the MFCC covariance features.

Finally, though we achieved an increased accuracy with our machine, **Figure 4** shows that the variability of even our most robust machine is high. For the false positive case, 50% of such calls occurred between 0.06 and 0.08, which is reasonable. However, 25% of false positive calls occurred for data between 0.01 and 0.04 TAC which is quite distant from the legal limit. In the false negative case, the variance of our machine is even more drastic – 25% of missed calls occur for data with TAC of 0.14 - 0.23 where patients would be considered extremely intoxicated. This variance underscores the fact that, though our system takes a step in the right direction for using mobile phones to classify sobriety, our tool is only intended for research classification tasks and is not meant to be implemented to identify drunkenness in contexts with social consequences. Using machine learning to discern between sober and intoxicated subjects outside of a research setting would require further algorithmic design with many more inputs than solely an accelerometer, and would require a far higher level of explainability than, say, an SVM provides on its own.

Future Work and Improvements

Though we have achieved a promising level of accuracy, our study was challenged by several factors. For instance, in our cohort we did not control for phone placement (pant pocket, shirt pocket, purse, etc.) which very likely added noise to our training. Additionally, we did not isolate for walking events before storing data. We attempted to extract walking data during post-processing, but found that this generally harmed classification accuracy. Using a mobile-device's built in algorithm for flagging walking data at sensor time might address this issue and potentially boost downstream accuracy. It is also important to note that all of our classifiers performed better when classifying sober data vs. intoxicated data and that the variance of our most robust classifiers was high. This suggests that there is still a fundamental lack in our understanding about what differentiates the two – further investigation would likely lead to improved results.

We plan to use the classifiers trained in this study to build a mobile application that is freely available to the public for voluntary users wishing to monitor their drinking habits or as a tool for researchers tracking sobriety in real-time without expensive sensors. Trained on a rich set of real-world field data with labels generated by unbiased sensors rather than self-reports, our classifiers would serve as one of the most robust accelerometer-only sobriety trackers available to the public. Using a combination of our machines to capitalize on their relative performances on each class of data, we could theoretically improve the accuracy of the system to be as high as 77%.

Conclusion

Herein we gathered a high-quality dataset suitable for training a machine learning classifier to differentiate between a sober and intoxicated subject using only tri-axial accelerometer signals. The dataset was unique, in that it was more reliable than most studies of its nature, given that ground truth intoxication levels were established using unbiased sensors, rather than self-reports. Further, on our dataset we achieved the highest known binary classification accuracy for accelerometer-only classification of sobriety, obtaining an average test accuracy of 75.04% with a support vector machine. We also identified a highly informative new set of features to be investigated in future studies; namely MFCC covariance matrix entries. These features produced an absolute gain in classification power of up to 8% in our study.

More work is needed to better understand what differentiates sober from intoxicated walking patterns – but these results will serve to improve the baseline for future studies addressing this issue. In the meantime, we plan to use the classifiers trained herein to build a mobile application that is freely available to researchers and the public to track sobriety in real-time using only mobile accelerometer signals.

Bibliography

1. Rehm J, Mathers C, Popova S, Thavorncharoensap M, Teerawattananon Y, Patra J. Global burden of disease and injury and economic cost attributable to alcohol use and alcohol-use disorders. Lancet. 2009;373(9682):2223-33.
2. Suffoletto B, Kristan J, Chung T, Jeong K, Fabio A, Monti P, et al. (2015) An Interactive Text Message Intervention to Reduce Binge Drinking in Young Adults: A Randomized Controlled Trial with 9-Month Outcomes. PLoS ONE 10(11): e0142877. DOI: https://doi.org/10.1371/journal.pone.0142877
3. Kao H-L C, Ho B-J, Lin AC, Chu H-H. Phone-based gait analysis to detect alcohol usage. Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp '12 (2012) 661 doi:10.1145/2370216.2370354.
4. Arnold Z, Larose D, Agu E. Smartphone inference of alcohol consumption levels from gait. Healthcare Informatics (ICHI), 2015 International Conference on. IEEE. 2015. 417-426. DOI: 10.1109/ICHI.2015.59.
5. Bae S, Ferreira D, Suffoletto B, Puyana JC, Kurtz R, Chung T, Dey AK. Detecting Drinking Episodes in Young Adults Using Smartphone-based Sensors. Conference: PACM Interact. Mob. Wearable Ubiquitous Technol. (IMWUT), At Maiu, Hawaii. 2017;1(2):5. DOI: 10.1145/3090051.
6. Gharani P, Suffoletto B, Chung T, Karimi HA. An artificial neural network for movement pattern analysis to estimate blood alcohol content level. Sensors 17 (12) (2017) 2897.
7. InfluxData. The Modern Engine for Metrics and Events. InfluxData, Inc. 2018. https://www.influxdata.com/
8. SCRAM Systems. Research: Transdermal. Alcohol Monitoring Systems, Inc. 2018. https://www.scramsystems.com/research/c/transdermal/
9. MathWorks. Documentation: cheby2. The MathWorks, Inc. 2018. https://www.mathworks.com/help/signal/ref/cheby2.html
10. Clapp J, Madden D, Mooney D, Dahlquist K. Examining the Social Ecology of a Bar-Crawl: An Exploratory Pilot Study. Submitted to PLoS one for review. 2017.
11. Pachi A, Ji T. Frequency and velocity of people walking. 2005. 83. 36-40.
12. Lamoth CJC , Ainsworth E, Polomski W, Houdijk H. Variability and stability analysis of walking of transfemoral amputees. Medical engineering & physics. 2010;32(9):1009-14. DOI: http://dx.doi.org/10.1016/j.medengphy.2010.07.001.
13. Abadi M, Agarwal A, Barham P, Brevdo E, et al. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. 2015. https://www.tensorflow.org/
14. Chollet F, et al. Keras. 2015. https://keras.io
15. Chang C-C, Lin C-J. LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. http://www.csie.ntu.edu.tw/~cjlin/libsvm
16. awjuliani/sound-cnn. A convolutional neural network that classifies sounds. Github, Inc. 2018. https://github.com/awjuliani/sound-cnn

17. Pedregosa F, et al. Scikit-learn: Machine Learning in Python. JMLR 12, pp. 2825-2830, 2011.