Do we dream or do we have a nightmare?

Joshua Antonson
Carnegie Mellon University
Department of Electrical and Computer Engineering
Pittsburgh, PA, USA
jantonso@andrew.cmu.edu

Abstract

Sleep is fundamental to the overall health and well being of humans. In this paper, I present a series of methods to analyze the sleep/wake pattern of a user through a simple android application. By recording accelerometer data during a night of sleep, I can identify user movement and periods of restlessness, when the user wakes up, and provide the user a prediction of their future sleep/wake patterns. Preliminary results confirm the ability to identify user movement and periods of restlessness, as well as when the user wakes up. I can successfully predict the user's sleep patterns by aggregating the data regarding each of their previous nights of sleep.

Introduction

Sleep is fundamental to the overall health and well-being of humans and is therefore a widely studied phenomenon. There are five stages of sleep: 1, 2, 3, 4, and REM (rapid eye movement) sleep. We cycle through the stages from 1 to REM, and then repeat. Stages 1 and 2 are classified as light sleep, stages 3 and 4 are referred to as deep sleep, and stage 5 is REM sleep. As mentioned above, when we fall asleep and how we sleep is largely affected by our behavior leading up to falling asleep. The more consistent our routine, the better we sleep. In addition, environmental factors, such as temperature, darkness, and quietness affect how we sleep. Being able to identify the sleep/wake behavior allows us to determine the restlessness of a user and their different sleep cycles. From this, the user can see their sleep habits and how well they are actually sleeping.

In order to analyze the sleep cycles of the user and their overall sleep, we must be able to identify the general movement of a user during sleep. A well-known method of doing so is to use accelerometer data collected by sensors during the course of sleep. My approach is to use an android application placed on the bed to record accelerometer data, which allows me to collect accurate data without imposing any new technology onto the user such as wearable sensors or sensors in the bed. From this, I can distinguish between the when the user lies still on the bed versus when they make a noticeable movement such as rolling over in bed. I then analyze the periodicity of the user's sleep movements and train a classifier in order to recognize different periods of sleep. This is very challenging using only movement related data, as a large piece of the different sleep stages is different bodily behavior such as varying brain waves, body temperature, and heart rate.

My approach accurately identifies movement events during sleep, recognizes when a user wakes up, is restless, or is asleep, and predicts their future sleep/wake patterns.

Previous work

I will briefly describe a few approaches to the problem of analyzing the sleep/wake pattern. In relation to my work, these fall into two categories: applications that monitor the sleep behavior of a user and algorithms/studies that analyze movement events and the different phases of sleep using accelerometer or other sensor data.

Most of the current sleep apps focus on waking the user up at the optimal time. Sleep Bot and Sleep Cycle are two popular sleep tracking phone applications. They use accelerometer and other movement related data to estimate the sleep phases of the user. The user can set a target time to wake up, and then the app will wake the user when they are in the light sleep phase and near the target time. They can also see a time based graph which displays each of the different sleep phases. This is closely related to my approach, but I am also focusing on collecting and aggregating data across users, as well as being able to predict the user's future sleep/wake patterns.

There are numerous algorithms and studies related to using movement data collected during sleep to estimate the sleep/wake pattern. I will briefly discuss one that relates closely to my approach. During this study, three professors at the University of Virginia collected movement data using the WISP platform: active RFID-based sensors equipped with accelerometers. They compared the results using these WISP sensors to both pressure sensors embedded in a mattress and to a simple iPhone application, which is similar to my method. Their method was 100% accurate in detecting discrete movement events during sleep and is 90% accurate at determining body position on a bed. This study is one of many in the field of using sensor data to analyze the sleep/wake patterns.

Approach

I built an Android application that collects accelerometer data during sleep. The user leaves the phone on their bed while they sleep and it records the accelerometer data during sleep. The accelerometer sensor records five measurements per second and records the x, y, and z acceleration measurements in m/s^2. Every minute, the application sends the data for the prior minute time period to a server running in Google App Engine, which stores it for processing at a later time. The accelerometer in a user's phone is not the most accurate method to measure movement patterns, especially since the phone is placed on the bed next to the user. A more accurate method would be to place sensors on the user or cover the mattress with an array of sensors, but by using the accelerometer within the phone I can create an inexpensive and unobtrusive method to collect reasonably accurate movement data.

Once the user wakes up, the app will display information to the user regarding their sleep/wake pattern via a time-based graph of movement events. Figure 1 shows an example of a user's time-based graph of movement events for a sample night of sleep. I will explain what the movement events correspond to shortly, but intuitively they capture the user's movements during a night of sleep.

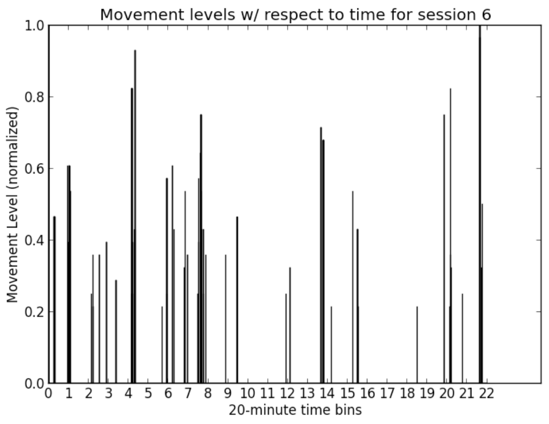

Figure 1

An initial training set of six users label key points during their sleep. The acceleration data is broken up into 20-minute time bins and the user labels each time bin as "restless", "woke up", or "asleep". A "restless" label corresponds to a period of time when they are attempting to fall asleep, possibly tossing and turning. A "woke up" label corresponds to a period of time when they woke up and an "asleep" label corresponds to a time period when they were asleep. Clearly, we are making a very large assumption that the user accurately labels their sleep. In addition, a user knows when they woke up, but they are essentially estimating when they fell asleep in order to distinguish between a "restless" label and an "asleep label", so we must keep that in mind when accounting for the accuracy of our approach. Figure 2 shows an example of the user's labeling for the previous time-based graph of movement events with "r' corresponding to "restless", "w" corresponding to "woke up", and "a" corresponding to "asleep".

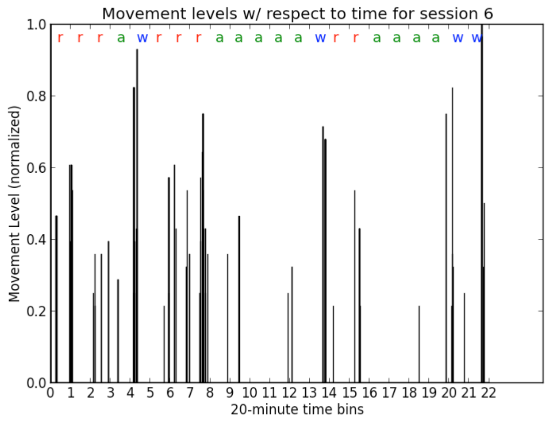

Figure 2

My approach consists of four key steps. The first is data collection, which was just discussed, and consists of gathering acceleration data from the user while they sleep. In addition, an initial training set of users label each 20-minute time window for each night of sleep over the course of a week. Second, from the acceleration data I can identify and extract movement events during a night of sleep for the user. Next, I can classify the sleep according to these movement events and the initial set of user's labels. Finally, I can predict the future sleep/wake patterns of the user by aggregating data from each of their previous nights of sleep.

In order to be able to classify the sleep/wake patterns of a user we need to be able to identify movement events. Intuitively a movement event corresponds to a movement made by the user while they sleep, i.e. rolling over in bed, sitting up, getting out of bed, etc. In order to identify and extract movement events we need to gather baseline accelerometer data that corresponds to no movement by the user. From this baseline data, we establish threshold accelerometer values for a given user's movements. I have the user place their phone on the bed and then have the user lay on their right side, left side, stomach, and back and also have the bed empty. For each of these sessions, the user stays still. From this, we gather accelerometer data related to no movement for the user in a wide range of different scenarios. The derivative of acceleration, for x, y, and z and with respect to time, stays within a certain threshold [-b1, +b1] for each of the respective scenarios. Of note, is the fact that the threshold for the x, y, and z accelerations can be different depending on the user's bed and phone. Figure 3 shows that the y acceleration threshold for a given user and no movement is [-0.08, 0.08] for a given scenario, i.e. phone on left middle of bed and user lying on right side.

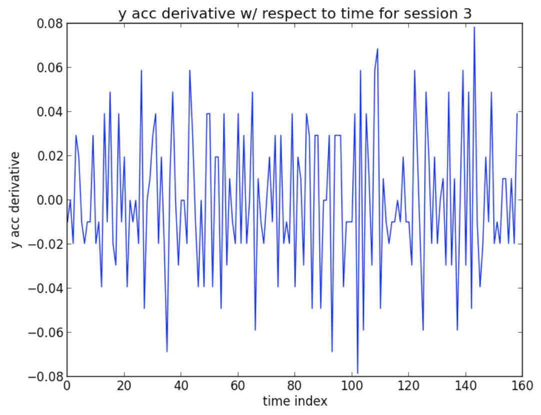

Figure 3

Once I have established x, y, z threshold values for no movement in each of the different scenarios for a user, I can identify movement events over the course of a night of sleep. I consider a movement event as when one of the acceleration derivative readings exceeds the threshold values established in the previous step. So, for example if the x acceleration derivative at a time step was 0.15 and the threshold was [-0.08, 0.08] as defined previously, then I would consider that time step as a movement event. For each two-second window, I define the movement weight of the window as the number of movement events during that window. So for each timestamp that the x, y, or z acceleration derivative exceeds the threshold within that two-second window, I increase the movement weight by one. Then, I normalize the weight of each window by the maximum weight of all the windows. We then remove all of the windows where the normalized weight is less than 0.20 as these weights correspond to subtle movements that do not affect sleep. I am only concerned with identifying large movements such as rolling over, sitting up, getting out of bed, etc. From this, we are left with two-second time windows that are weighted according to the amount of movement during that time frame

Next, I cluster windows of movement in order to identify twenty-second periods with multiple movements. If there are two or more two-second windows of movement within a twenty-second window, then I assign the ten two-second windows within that twenty-second window to have the weight of the maximum two-second period weight. From this, we have identified periods of clustered movement events throughout the user's sleep. Figure 1 from the previous page shows the movement levels (normalized) over the course of a 10-hour period of sleep for a user and is what the user sees once they are finished with a night of sleep, as mentioned previously.

The third key step is to classify the user's sleep. In this step we attempt to label each of the 20-minute time bins as "restless", "asleep", or "woke up". We train a classifier using 20-minute time bin of movement events and the corresponding user generated labels from a training set of data. The training set of data comprises one week of sleep for multiple users. I have tried three different algorithms to classify the user's sleep, each with two different methods of compressing the 20-minute time windows into more manageable sized number of features.

The first method of compression is to compress each 20-minute time bin of movement events, which has 600 movement levels data points, into a 4-tuple of four defining metrics. The four metrics are the mean of the movement levels, the maximum movement level, the number of non-zero movement levels, and the length of the biggest consecutive block of non-zero movement levels, all normalized.

The second method of compression is to compress each 20-minute time bin of movement events using the discrete wavelet transform. The discrete wavelet transform is similar to Fourier transforms, but has the advantage that it captures both the frequency and location information of the time signal. The signal is repeatedly passed through both a low pass filter and a high pass filter to get the approximation coefficients and the detail coefficients respectively. At each step, the time resolution of the signal has been cut in half, but the frequency resolution has been doubled. For my approach, I have used an 8-level dwt and then use the approximation coefficients as the compressed version of the 20-minute time window. More information on the discrete wavelet transform can be found at [6].

The first algorithm is k-NN with k = 1. The k-NN algorithm relies on a similarity measurement, so the problem boils down to defining a metric to measure the similarity between 20-minute time bins of movement events. For each new 20-minute time bin, I find its closest neighbor using the similarity metric and assign it to the label of its neighbor. Currently, I have used three different methods for determining the similarity of 20-minute time bins of movement events.

The first method is called dynamic time warping. Dynamic time warping attempts to find the optimal non-linear alignment between two time series. It essentially shifts one time-series over one step at a time and computes the Euclidean distance between the two time series for each alignment. The similarity measure for the two time series is then the minimum Euclidean distance, i.e. the optimal non-linear alignment. The drawback to this method is that the algorithm is quadratic in the length of the time series, but it can be sped up using a locality constraint. The assumption of the locality constraint is that it is very unlikely for two time steps in the respective time series, ti and tj, to be matched if i and j are very far apart, as determined by a window size w. Thus, it only needs to consider mappings that are within a given window size w. Using the LB Keogh lower bound of dynamic time warping can further optimize the algorithm, as it is an initial check to rule out alignments in linear time. More information on dynamic time warping can be found at [5].

The second and third methods for defining the similarity measurement is to use the two compression methods defined above with the Euclidean distance between the respective compressed versions as the similarity measurement.

The second algorithm to classify the user's sleep is using support vector machines. Given a set of training samples, each marked as one of two categories; a support vector machine builds a model that assigns new data points into one of the categories. It is referred to as a non-probabilistic linear classifier. Since we are trying to classify new 20-minute time windows as one of three categories: "restless", "woke up", or "asleep", our algorithm trains a classifier for each of the pairs of labels. It then aggregates the results of each of these three classifier's prediction for a new 20-minute time window in order to determine which is the best prediction for a new label.

The third algorithm to classify the user's sleep is the k-means clustering. The k-means clustering algorithm aims to partition the set of all samples into k clusters, assigning each observation to the cluster with the nearest mean. Since we have three different labels: "restless", "woke up", and "asleep", I have used k-means clustering with a k of 3.

Finally, we use the trained classifier to label each 20-minute time bin for the future sleep of a user as "restless", "woke up", or "asleep" in order to give them a better understanding of their sleep/wake pattern. I have tested each of the three classifiers and have chosen to use the support vector machines algorithm for classification, which I will discuss in the next section.

The final step in my approach is to predict the user's future sleep/wake patterns. This step consists of aggregating data from each of the user's previous nights of sleep. For each of the labeled 20-minute time windows, I assign a weight of 0.0 if it was labeled as "asleep", 0.5 if it was labeled as "restless", and 4.0 if it was labeled as "woke up". I then sum across each index for the 20-minute time windows and divide by the number of 20-minute time windows in that index, i.e. sum up the weights of the i-th 20-minute time windows for each night of sleep and then divide by the number of i-th 20-minute time windows. From this, we can display the aggregated weights for each 20-minute time window index to the user, which presents a good representation of their typical sleep/wake behavior, and intuitively a good prediction of their future sleep/wake pattern. Figure 4 shows a prediction for one of the users using 14 nights worth of data.

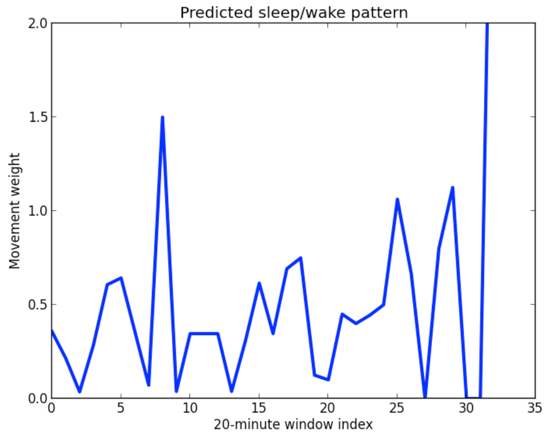

Figure 4

Experimental setup and results

So far, I have collected data using the application myself and five other users. I have collected data from a total of 31 nights worth of sleep. Figure 5 shows the distribution of the x, y, and z acceleration data for one night, chosen at random, from each of the users.

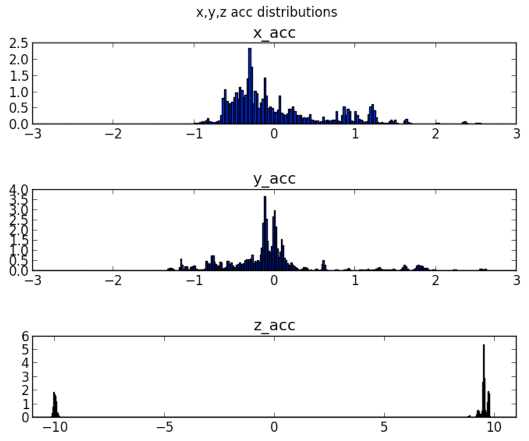

Figure 5

Intuitively, we can see that the x-acc distribution and y-acc distribution look to be Gaussian, and the z-acc distribution is mixed Gaussian with one centered on 9.8 and one centered around -9.8.

Figure 6 shows the x-acc distribution with the corresponding Gaussian fit with mean of -0.0185 and standard deviation of 0.886.

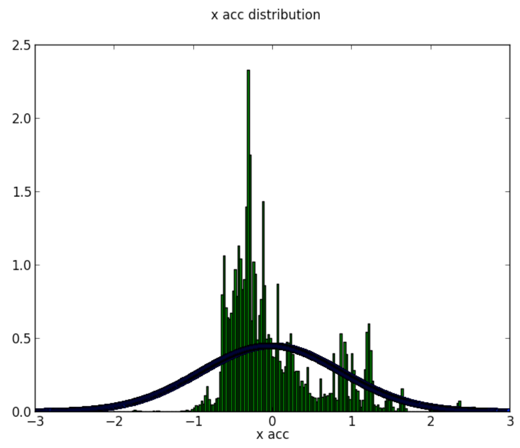

Figure 6

Figure 7 shows the y-acc distribution with the corresponding Gaussian fit with mean of -0.0188 and standard deviation of 0.706.

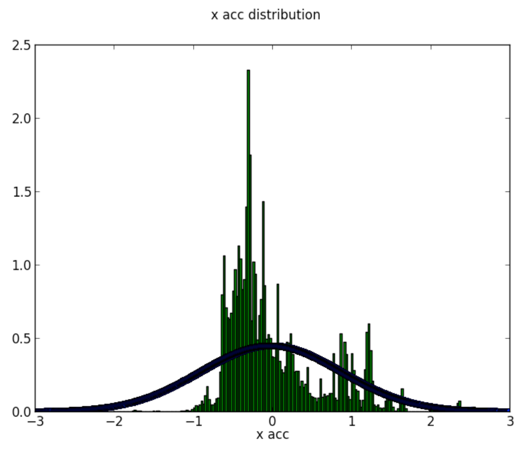

Figure 7

The Gaussian fits for the x and y accelerations have very large standard deviations with a mean right around 0.0. This is because when the user is lying still on their bed, the acceleration value is right around 0.0. When they make a movement, the value spikes up to between 1.0 and 2.0 or -1.0 and -2.0. Since the majority of the time the user is lying still, there are a lot more values close to 0.0 than above 1.0, hence the mean close to 0.0 but the large standard deviation.

Figure 8 shows the two populations for the z-acc distribution with the corresponding Gaussian fits with mean of -9.97 and standard deviation of 0.06 for the left population and mean of 9.56 and standard deviation of 0.14 for the right population.

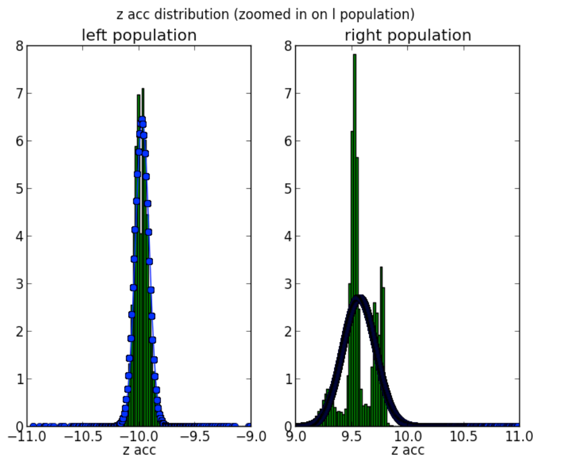

Figure 8

We are collecting acceleration data, so it makes sense to see the z-acc distribution be centered on 9.8, as that is the value of acceleration due to gravity. I believe that we are seeing the two populations centered on -9.8 and 9.8, respectively, because the orientation of acceleration data is unique for each phone. For the z-axis, which is perpendicular to the bed, we expect the value to be fairly consistent around 9.8 or -9.8, depending on the phone, as the likelihood of the phone flipping over mid-sleep is very low. This also explains why the standard deviation is a lot less for the z-acc distributions then the x and y distributions, as changes in z-acc are much less due to the z-axis orientation.

Figure 9 shows the distribution of acceleration data for myself across 7 nights of sleep.

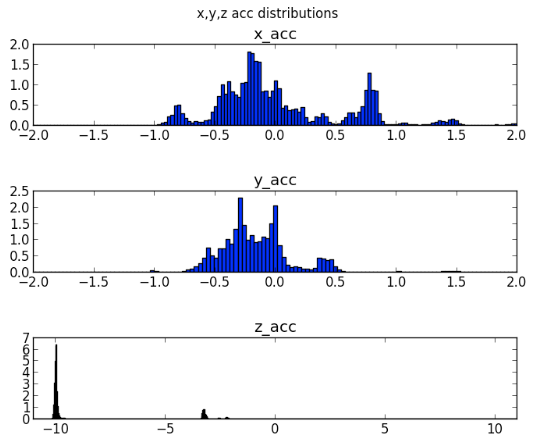

Figure 9

Figure 10 shows the distribution of acceleration data for a different user across 7 nights of sleep.

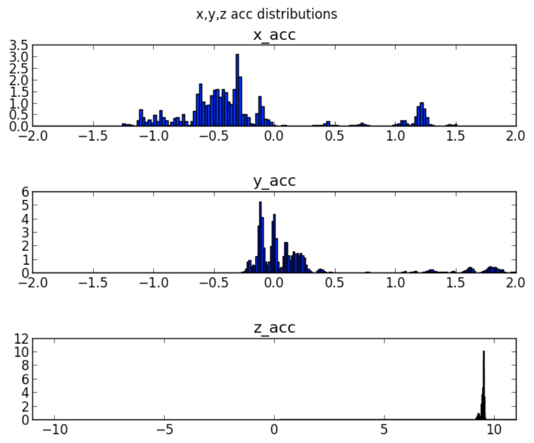

Figure 10

Comparing the distributions across two different users highlights the subtle differences in each user's accelerometer within their phone and sleep conditions. For example, the second user's z-acc is centered on 9.8, whereas mine is centered on -9.8, which confirms the hypothesis previously discussed. This emphasizes the need for a robust algorithm to detect movements from the acceleration data that is not uniform across users.

In addition, I have had each of the six users record five 10-second baseline sessions, where they lay still on their bed in each position: on left side, on right side, on back, on stomach, and empty bed. From this, I established the threshold ranges for x, y, and z acceleration derivatives that are used to identify movements, as discussed in the approach section.

In order to test the accuracy of my approaches identification of movement events, I have had each user record four 15-second sessions, where they lay still at the start then make a noticeable movement in the middle, such as rolling over, and then lay still at the end. Figure 11 shows the x, y, and z acceleration derivatives with respect to time for one of these sessions, i.e. rolling over from left side to right side.

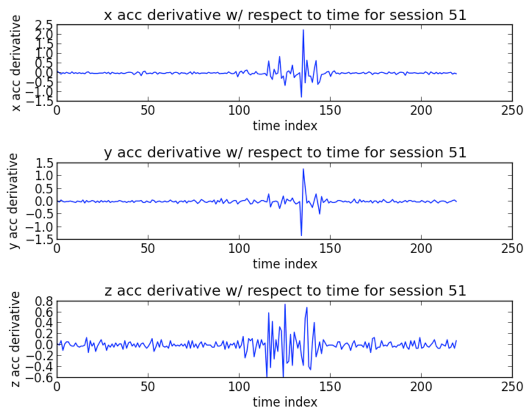

Figure 11

As you can see in Figure 11, there is one distinct spike in the x, y, and z acceleration derivative data that corresponds to the user's movement. Figure 12 shows my algorithm's attempt to identify the movement levels for each two-second window of this session. Each bar corresponds to the movement level (normalized) for each two-second window.

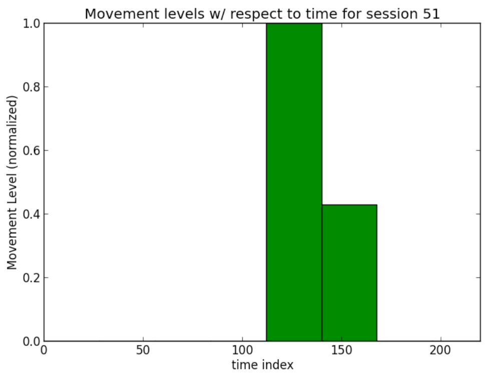

Figure 12

Figure 13 shows the x, y, and z acceleration derivative data overlaid on top of my algorithm's identification of movement levels for each two-second window of this session. Note: the acceleration derivative is absolute value in this figure.

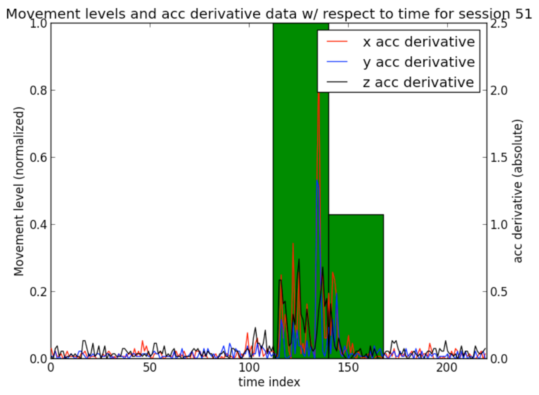

Figure 13

As you can see from Figure 13, my algorithm successfully identifies the one movement event, as the movement levels spikes when the acceleration data spikes during the user's movement and is 0.0 when the acceleration data corresponds to no movement.

Out of the 24 of these single movement sessions, my approach correctly identified 22 movements for an overall accuracy of 91.66% and didn't report any false movements. From this, we can reasonably assume the accuracy of identifying movements during a user's sleep.

Now, I will discuss the results that I have obtained related to the classification of labeling the user's sleep. There are 670 20-minute time bins of movement events from all six users. I have split up each night of sleep from the six users into two sets: training and testing. Training consists of a total of 19 nights worth of data and 390 20-minute time bins of movement events. Testing consists of a total of 13 nights worth of data and 280 20-minute time bins of movement events. Table 1 shows the classification results for each of the three classification algorithms: k-NN, support vector machines, and k-means clustering for the 4-tuple method of compression.

	precision	recall	f1-score
kNN	0.84	0.81	0.82
svm	0.82	0.84	0.82
kmeans	0.81	0.69	0.73

Table 1

The kNN algorithm classified "asleep" labels with 92.1% accuracy, "restless" labels with 32.4% accuracy, and "woke up" labels with 55.55% accuracy.

The svm algorithm classified "asleep" labels with 96.76% accuracy, "restless" labels with 37.8% accuracy, and "woke up" labels with 44.44% accuracy.

The kmeans-clustering algorithm classified "asleep" labels with 96.76% accuracy, "restless" labels with 59.45% accuracy and "woke up" labels with 7.4% accuracy.

Table 2 shows the classification results for each of the three classification algorithms: k-NN, support vector machines, and k-means clustering for the discrete wavelet transform method of compression.

	precision	recall	f1-score
kNN	0.81	0.80	0.79
svm	0.79	0.80	0.76
kmeans	0.75	0.79	0.71

Table 2

The kNN algorithm classified "asleep" labels with 91.2% accuracy, "restless" labels with 27.0% accuracy, and "woke up" labels with 59.26% accuracy.

The svm algorithm classified "asleep" labels with 98.14% accuracy, "restless" labels with 16.2% accuracy, and "woke up" labels with 22.2% accuracy.

The kmeans-clustering algorithm classified "asleep" labels with 99.53% accuracy, "restless" labels with 10.8% accuracy and "woke up" labels with 3.7% accuracy.

In addition, Table 3 shows the classification results for the k-NN classification algorithm using dynamic time warping.

	precision	recall	f1-score
kNN	0.40	0.44	0.41

Table 3

The kNN algorithm using dynamic time warping classified "asleep" labels with 64.7% accuracy, "restless" labels with 8.33% accuracy, and "woke up" labels with 50% accuracy. Clearly, the k-NN classification using dynamic time warping isn't accurate enough as compared to any of the other methods of classification. In addition, its runtime is on average 32.5 minutes, whereas each of the other methods runs on the order of 100 milliseconds.

From this, we can see that of the two compression techniques, using the 4-tuple method of compression is more accurate than the discrete wavelet transform method. Although, the overall precision, recall, and f1-score didn't decrease by too significant of percentages, once we look at the individual accuracies for each of the three labels we can see that the 4-tuple method of compression resulted in better accuracy across the board for each of the three algorithms.

In addition, of the three algorithms for classification, the kNN algorithm and svm algorithm had higher overall precision, recall, and f1-scores than the kmeans-clustering algorithm using the 4-tuple method of compression. We can see that the svm algorithm and kNN algorithm are very similar in terms of accuracy, but the svm algorithm is slightly more accurate when looking at individual labels, with only a slight decrease in the accuracy of "woke up" labels.

Because of this, I have decided to use the svm algorithm with the 4-tuple method of compression to classify the 20-minute time windows. Also, the code for the svm algorithm is much easier to implement and is a lot cleaner than the kNN algorithm.

Clearly, the main limitation is the accuracy of each classifier in labeling "restless" and "woke up" labels. Each of the three classifiers is very accurate at labeling "asleep" labels, but has accuracies near or below 50% when labeling "restless" or "woke up" labels. I believe that this hints at an underlying issue in my approach, which is that I haven't found a significant difference in movement behavior between a 20-minute window labeled as "restless" versus one labeled as "woke up". This makes sense, as intuitively the movement of a user who woke up is very similar to one who is restless with multiple noticeable movements in both cases. Thus, my classifier is having a hard time deciding between "restless" or "woke up" labels for 20-minute windows that correspond to either a user waking up or being restless, i.e. labeling a "restless" period as "woke up" or vice versa. In order to combat this, I have tried to impose spatial constraints on the classifier, meaning that if a predicted "restless" label is preceded in time by a predicted "asleep" label, then I switch the "restless" label to a "woke up" label because intuitively a user must wake up, before they can be restless. If I have more time, I would look into potentially training a classifier to predict between two labels, “asleep” or “not asleep”. From this, we could then label each first “not asleep” label as “woke up” and any preceding “not asleep” labels as “restless” in a consecutive block of “not asleep” labels. This should ideally increase the accuracy of my classifier, as distinguishing between “asleep” and “not asleep” labels is a lot easier, as the movement behavior in the respective 20-minute time windows is significantly different.

Conclusion and future work

So far, I have successfully implemented an algorithm that reliably detects movement events. In addition, I have tested out three different classification methods and two compression methods for recognizing sleep labels. I have discovered that the svm classifier using 4-tuple method of compression is the most accurate. Using this method my classifier can detect user labels with 84% accuracy using a training and test set comprising six users and 31 nights worth of acceleration data and user-generated labels, but still needs to be improved in distinguishing between “restless” and “woke up” labels. Also, I have created a simple and accurate method for predicting the future sleep/wake patterns of a user given the aggregated results of their previous nights of sleep.

My approach is scalable for multiple users, as the only limitation would be the size and performance capabilities of the servers, as the server handles the movement identification and label classification. Since, each of these is largely independent work across multiple users, we can scale to multiple users by simply scaling size and performance of the servers. In addition, more user data will only increase the performance of my classification method.

In the future, I would like to look into using the user’s predicted sleep pattern and their labeled periods of sleep in order to label the stages of sleep for each user. In addition, improve the robustness of the classifier for distinguishing between “woke up” and “restless” labels, as mentioned previously. Finally, I could incorporate the “smart” alarm clock feature that the majority of other sleep applications provide to users.

The android application easily allows users to record their sleep and accurately identify their sleep/wake patterns. To the best of my knowledge, no other sleep-recording app combines data across users in order to predict sleep. By leveraging the network of users using the application, I am able to provide accurate metrics that will help users understand their sleep and gather valuable data to help understand the key trends in human sleep.

Lessons learned

Overall, I learned a great deal from this project. This was my first real experience with machine learning in actual practice, not just understanding the theoretical basis of it. I definitely underestimated how hard it would be to develop a classifier to predict labels for users sleep that would be accurate enough to be actually used in production. Also, I just assumed that there would be enough of a significant difference in the features between “asleep”, “restless”, and “woke up” labels, where in actuality I don’t think there is too significant of a difference between “restless” and “woke up”, as I discussed prior, which I believe led to the inaccuracy of my classifier. In addition, this was my first experience working on a project that would be considered big data, so I ran into certain issues associated with that. For example, certain algorithms would take much longer to run then I expected. Also, I constantly ran into storage limit issues with Google App Engine, which made my project a lot harder than I had anticipated.

REFERENCES

[1] Crean, Dan. *Sleepdex*. 27 April, 2014. <http://www.sleepdex.org/about.htm>

[2] Hong, Charles Chong-Hwa, et al. “REM sleep movement counts correlate with visual imagery in dreaming: A pilot study”.

[3] Hoque, Enamul, et. al “Monitoring Body Positions and Movements During Sleep using WISPs”. <http://www.cs.virginia.edu/~stankovic/psfiles/sleep.pdf>

[4] Owens, Judith, et al. “Television-viewing Habits and Sleep Disturbance in School Children.” *Pediatrics* Vol. 104 No. 3 (1999). American Academy of Pediatrics. 2013<http://pediatrics.aappublications.org/content/104/3/e27.full>

[5] Minnaar, Alex. “Time Series Classification and Clustering” <http://nbviewer.ipython.org/github/alexminaar/time-series-classification-and-clustering/blob/master/Time Series Classification and Clustering.ipynb>

[6] K., Parashar. “Discrete Wavelet Transform” <http://www.thepolygoners.com/tutorials/dwavelet/DWTTut.html>