Actibetes: An Activity Recommender System for Patients with Diabetes - Transportation mode detection

Tim Raulf-Pick,
Department of Electrical and Electronic Engineering, Imperial College London, SW7 2AZ
Email: tcr12 @ic.ac.uk
Supervisor: Dr. Yiannis Demiris

***Abstract*—Diabetes remains a serious long-term complication where a sustained high blood sugar level is experienced. Prevention, as well as treatment, revolve around a healthy diet, physical exercise and maintaining a normal body weight. The following paper outlines the use of a mobile application, specifically the use of sensor metrics (GPS, accelerometer etc.) to aid the application in making informed decisions for patients.**

I. Introduction

Over 387 million people are affected by diabetes worldwide [**?**] which represents a striking 8.3% of the world's population. Considering its ability to more than double a person's fatality risk, we treat diabetes as a very serious condition. An important aspect in the proper treatment and management of diabetes is the correct maintenance of glucose levels throughout the day. Several factors play a key role in doing so, including a certain amount of physical activity. The correct amount of exercise depends both on current glucose levels as well as the individual's reaction and lag-effect to physical exercise. With an increased cost of diabetes in the coming decades, alongside alarming rising obesity rates [**?**], the usage of activity data will become more prevalent for the correct treatment of diabetes.

Smartphones provide a unique tool for integrating health monitoring in a patient's daily life. The concept behind Actibetes is to provide user tailored suggestions to help maintain adequate glucose levels through correlating the patient's blood sugar level fluctuations with corresponding levels of physical activity. As every individual has a very specific reaction to physical exertion, our system will be able to leverage the application's machine learning features to provide customised feedback. Actibetes uses the patient's location and the in-built iPhone accelerometer/gyroscope to gather and accumulate information regarding the levels of physical activity achieved by the user.

II. Background

Location services have been a tool in Apple's iOS from the very first days of the iPhone. Up until the iPhone 3G location was only to be established through triangulation from cellular towers and Wi-Fi networks as no physical GPS hardware chip was embedded in the device. More recent iPhones have been able to leverage a physical GPS chip for a multitude of purposes in the iPhone. The concept behind GPS triangulation is to receive a fix from four different satellites and use mathematical extrapolation to determine their positions relative to the receiver. Four satellites are required, three for position triangulation and one for true time deviation.

The iPhone provides several other sensors which may become of interest for Actibetes. Primarily the accelerometer can provide crucial movement information in the x, y and z plane allowing us to calculate general levels of movements as well as steps taken by the user (pedometer). Together with another onboard sensor, the gyroscope, one can depict a very accurate image of movement of the device.

III. Requirements

The requirements of this sub-component is to provide details of the patients activity levels throughout the day to feed into the recommender model to help create personalised recommendations as to how the user can best regulate his/her glucose levels. Hence classifiable energy bands (i.e. sitting, walking, running) are required, alongside time information to feed into the recommender system. Furthermore the system needs to be able to identify between self-propelled motion versus sitting in a bus or car.

To aid in correctly feeding the recommender with activity data we must attempt at filtering out the user cheating his exercise levels [1]. This will require the algorithm to be able to distinguish between the user shaking the device in his hand rather than actually moving.

IV. Using the accelerometer to predict motion type

The accelerometer and gyroscope outputs data in the x, y and z axis which in itself is only useful in certain specific circumstances. Clearly motion isn't uniquely defined to a specific axis and hence it is difficult to identify specific motions based on singular axis information. An exception to this applies in identifying a stationary standing individual where the x and z axis will show little acceleration whilst the y axis will display a value in the area of $9.8m/s$ due to the acceleration of gravity. To make the data provided more stable and useful it is common practise to define the magnitude of the accelerometer data instead [2][3][4].

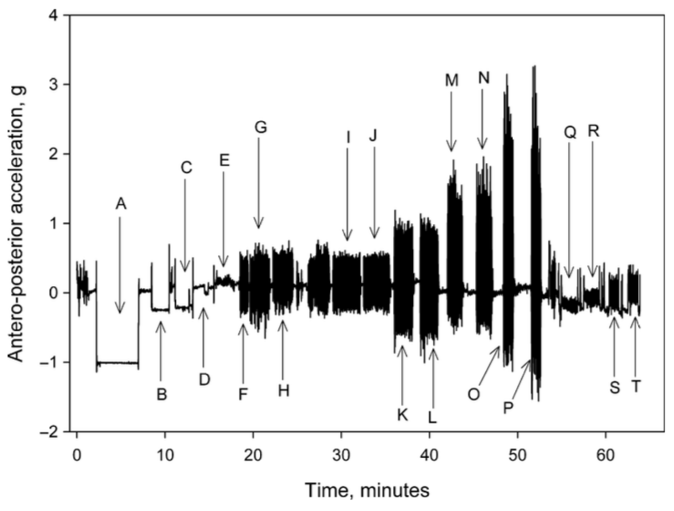


Fig. 1. Acceleration measured in the z-axis. Arrows highlight the signal measured during the tasks included in the test: lying (A); sitting (B); sitting on a computer (C); standing (D); washing dishes (E); walking along a corridor (F); walking downstairs (G); walking upstairs (H); walking outdoor (I-L); running outdoor (M-P); cycling outdoor (Q-T) [6].

$$m = \sqrt{(V_{x,s}^2) + (V_{y,s}^2) + (V_{z,s}^2)}$$

For each of the magnitudes a specific time stamp is provided as well giving a data structure as follows,

$$< t, V_{x,s}, V_{y,s}, V_{z,s} > .$$

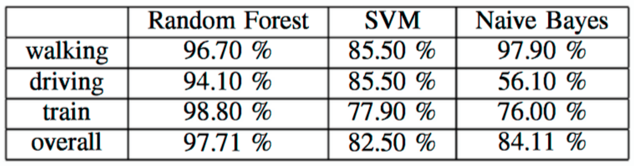
A. Feature extraction

The next step is to extract specific features in the magnitude and hence use these for further data analysis [5]. Looking at figure 1 one can see the differences in acceleration magnitude dependent on various types of movement and exercise. Figure 1 provides a level of granularity beyond that required in this project but the main key exercise types are present, including laying down, sitting, walking, running and biking. It can also be seen how using simple feature extraction it may become difficult to differentiate between walking and cycling as they both provide similar accelerometer magnitude readings.

There are three main types of feature extraction that become relevant for the purpose of identifying motion [7]. Frame-based feature extraction allows the algorithm to identify statistical features (e.g. mean, variance and kurtosis) as well as time domain metrics (e.g. double integral and auto-correlation) [7]. Using these against of a set of reference data for the various activity types will provide a good platform for the classifying algorithms. Peak-based feature extraction is another useful method at trying to identify prevalent information [8]. Especially for low frequency signals, such as those of break periods in a car, or slight deceleration in a train [6]. Looking at time between "peak areas" can help identify periods of acceleration change. Kinetic activities are classified as being high frequency [6], whilst stationary and motorised movements are mainly low-frequency. The third method of feature extraction, segment-based feature extraction observes and characterises patterns of acceleration over the given segment period. Analysing the variance of individual peak-based features allows for detection of velocity change [9]. This provides a third alternate feature extraction method, mostly used in conjunction with the other two.

B. Data Classification

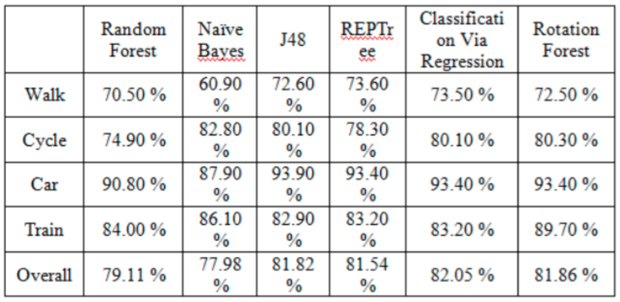
Data classification is an important aspect of the algorithm and in simple terms can be defined as a motion type classifier that takes as input a set of features F_k^* for a sequence k and a feature training set T_F, and produces as output a value $c(F_k^*, T_F) \in$ that represents the estimated motion type, i.e. m^W, m^D or m^T [10]. The three main types of algorithms used for classification are random forest [11], support vector machines [12] and Naive Bayes [13].
The Random Forest algorithm searches through a node network of decision trees which represent certain features of the input signal (in this case the accelerometer magnitude) and hence classifies a specific output node as that for running, walking or driving [10]. The Support Vector Machine (SVM) first introduced by [12] is used in modern classification analysis. SVM maps the data points in input on a plane, separates the positive and negative instances using a separation gap, and classifies the new instances based on which side of the gap they fall on [10][14]. Finally, the Naive Bayes is a probabilistic-based classification algorithm [13]. In simple terms each classification hypothesis is assigned a posteriori-probability based on the observed data set [10].



	Random Forest	SVM	Naive Bayes
walking	96.70 %	85.50 %	97.90 %
driving	94.10 %	85.50 %	56.10 %
train	98.80 %	77.90 %	76.00 %
overall	97.71 %	82.50 %	84.11 %

Fig. 2. Accuracy of different algorithms without biking [10]

Figure 2 shows the results of the algorithms discussed (using Waikato Environment for Knowledge Analysis WEKA) without the inclusion of biking as an activity. As previously explained biking shows similar features to those of walking and hence can create difficulty for the classification algorithm. The Random Forest algorithm shows the best overall performance whilst the Naive Bayes algorithm performs well in an environment where no two classification classes are of similar feature type. This can be further seen in figure 3 where walking and biking display similar feature sets and hence cause a large drop in the overall accuracy of the Naive Bayes algorithm. The ability to identify walking has dropped from 97.9% to a startling 60.9%.



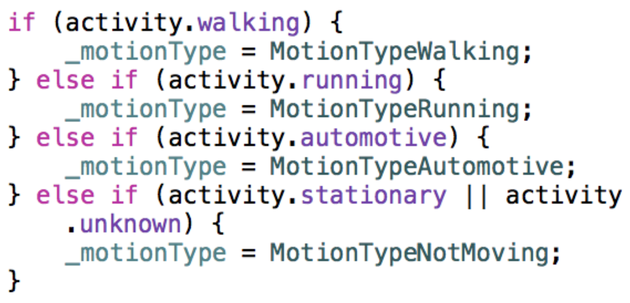
	Random Forest	Naïve Bayes	J48	REPTree	Classification Via Regression	Rotation Forest
Walk	70.50 %	60.90 %	72.60 %	73.60 %	73.50 %	72.50 %
Cycle	74.90 %	82.80 %	80.10 %	78.30 %	80.10 %	80.30 %
Car	90.80 %	87.90 %	93.90 %	93.40 %	93.40 %	93.40 %
Train	84.00 %	86.10 %	82.90 %	83.20 %	83.20 %	89.70 %
Overall	79.11 %	77.98 %	81.82 %	81.54 %	82.05 %	81.86 %

Fig. 3. Accuracy of different algorithms with biking [2]

The Random Forest algorithm also suffers an accuracy loss, but remains at the top-end of the previously discussed algorithms. In a paper by [10] it was suggested to introduce more complex algorithms such as the rotation forest [15], which may prove more accurate as it promotes both individual accuracy and diversity within the ensemble. Looking back at figure 3 we can establish that indeed the rotation forest algorithm provides the highest accuracy for the close-feature data-set.

V. Implementation

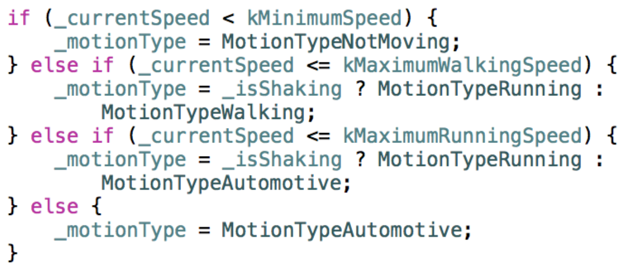
For the use in our prototype we will utilise both the functions provided by the accelerometer/gyroscope as well as the in-built GPS chip. Using Apple's Core Motion framework [16] is a convenient method at implementing the above discussed activity detection algorithm. The use of *CMMotionActivity* provides a boolean return stating wether the device is currently stationary, walking, running, automotive or unknown. This uses the proprietary Apple algorithms to identify motion type based on the accelerometer data. To limit the effect of previously discussed limitations in the use of the accelerometer our prototype uses the GPS chip to cross-reference the devices speed to further increase the accuracy of the prediction.



```
if (activity.walking) {
    _motionType = MotionTypeWalking;
} else if (activity.running) {
    _motionType = MotionTypeRunning;
} else if (activity.automotive) {
    _motionType = MotionTypeAutomotive;
} else if (activity.stationary || activity
    .unknown) {
    _motionType = MotionTypeNotMoving;
}
```

Fig. 4. Code Implementation of CMMotionActivity [17]

Figure 4 shows the if-else statement to categorise the motion type based on the output of the *CMMotionActivity* boolean. Both stationary and unknown activity types are classified as not moving in this implementation. As explained, to increase the reliability of this result figure 5 shows how this motion type is cross referenced against *_currentspeed* [18][19]. Several variables needed definition to classify the activity based on speed including, *kMinimumSpeed*, *kMaximumWalkingSpeed* and *kMaximumRunningSpeed* [17]. This allows customisable parameters to be used to calculate what type of activity is being conducted, this function is called in any instance where the device is in reception of a GPS signal.

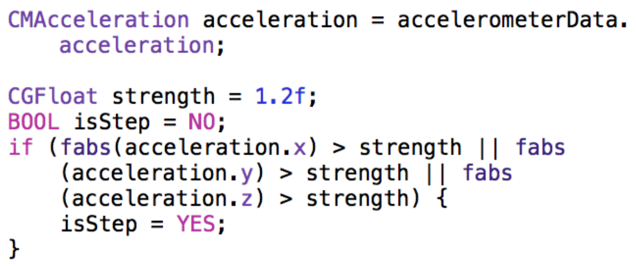


```
if (_currentSpeed < kMinimumSpeed) {
    _motionType = MotionTypeNotMoving;
} else if (_currentSpeed <= kMaximumWalkingSpeed) {
    _motionType = _isShaking ? MotionTypeRunning :
        MotionTypeWalking;
} else if (_currentSpeed <= kMaximumRunningSpeed) {
    _motionType = _isShaking ? MotionTypeRunning :
        MotionTypeAutomotive;
} else {
    _motionType = MotionTypeAutomotive;
}
```

Fig. 5. Code Implementation of GPS cross-reference

A. Identying steps

The accelerometer also provides the capability of counting the steps and this is utilised in our prototype as a further measure (alongside time) to quantise the amount of exercise by the patient. Using the inbuilt *CMAcceleration* function provided in the accelerometer data class [16] we are able to retrieve the acceleration in the x, y and z direction. Using an IF statement to identify if the absolute value of the acceleration in any given axis is greater than a given threshold will increase the step counter accordingly [17]. Implementation is seen in figure 6.



```
CMAcceleration acceleration = accelerometerData.
    acceleration;

CGFloat strength = 1.2f;
BOOL isStep = NO;
if (fabs(acceleration.x) > strength || fabs
    (acceleration.y) > strength || fabs
    (acceleration.z) > strength) {
    isStep = YES;
}
```

Fig. 6. Code Implementation of step-counter

B. Identiying cheating

The idea here is to avoid patients from cheating their activity measurements and hence receiving false outputs from the recommender model. Since the application is operating in the healthcare space this can prove to be quite an important safeguard as a patients well-being is at stake. The basis of identifying cheating in the prototype is by doing a check for any abrupt movements in a very short time frame. In other words, in a very short time segment the accelerometer will check for any large acceleration which are larger than even the smallest accelerations from running. The premise her is that if the time segment is small enough any accelerations of a human stride would not be large enough and hence only shaking by the hand (much smaller directional displacement) is likely. The algorithm then updates the status to *shaking* and disregards the result.

C. Interfacing with the rest of the system

The prototype solves the basic problem of monitoring and reporting the users activity levels throughout the day classifying them according to exercise intensity and keeping a record of the associated time. Further to this it attempts at filtering out users trying to falsify results and discredits the corresponding results. The return of the sub-system to the application will be in the form,

$$< t_{start}, t_{end}, activity\_type, cheated > .$$

This allows the recommender to feed in a constant stream of physical exertion levels of the patient into the recommender. Any results which are classified with a $true$ boolean for cheating will result in stationary exercise levels (i.e. the patient being seated). The actual processing will be done in the back-end servers hence it is important that our application can seamlessly transfer this information from the sub-system through the application into the back-end server.

D. Further Extensions

An interesting extension of the system would be to provide a higher level of granularity into the ability to identify different levels of exercise. It is yet to be decided how many levels of exercise we are going to use for the recommender system (depends mainly on the level of granularity available in glucose readings) and if possible matching a more accurate scale of physical exertion of the patient would provide stronger recommender outputs. To do this it would be possible to use the GPS data, more precisely the speed of movement, to give a more accurate prediction wether the user is walking (quickly or slowly) or running (quickly or slowly).

Furthermore, improvements in the reliability of the accelerometer data may help in receiving more accurate readings regarding activity type, which is especially important in environments where the assistance of the GPS is not available. Another room for thought may be around allowing the user to vet his activities, as well as customising them. Allowing interaction with the user through the front-end application as to approving the activity level given by the application may allow for a more appropriate and correct understanding of the patients daily activities. This addition would also hold the further benefit of allowing the user to input his custom activities into the recommender system which cannot be automatically detected by the device. Such activities could include rowing, yoga, fitness and football.

VI. Conclusion

Detecting motion and motion type has become a much discussed topic in recent years and has very useful applications in the space of diabetes and patient support applications [20]. The correct identification of activity type and level of endurance is paramount as life changing medical advise is made on the basis of the output. It is therefore important to ensure that all information is accurately analysed and processed. The above stated algorithms make a good attempt at utilising all available sensory inputs in the iPhone to build and accurate and reliable prediction of the users activity level. The use of GPS avoids misinterpreting fast acceleration for running in a motorised environment, whilst the accelerometer data when analysed in small time segments can be used to identify the user shaking the device, and hence cheating.

References

[1] M. Tomlein, P. Bielik, P. Krátky, S. Mitrík, M. Barla, and M. Bieliková, "Advanced pedometer for smartphone-based activity tracking.," in *HEALTHINF*, pp. 401–404, 2012.

[2] A. Thakore and S. P. Soni, "Enhanced user motion detection using multiple smartphone sensors,"

[3] S. Das, L. Green, B. Perez, M. Murphy, and A. Perring, "Detecting user activities using the accelerometer on android smartphones," *The Team for Research in Ubiquitous Secure Technology, TRUSTREU Carnefie Mellon University*, pp. 1–10, 2010.

[4] A. Allan, *Basic Sensors in iOS: Programming the Accelerometer, Gyroscope, and More.* " O'Reilly Media, Inc.", 2011.

[5] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[6] A. G. Bonomi, A. Goris, B. Yin, and K. R. Westerterp, "Detection of type, duration, and intensity of physical activity using an accelerometer," *Med Sci Sports Exerc*, vol. 41, no. 9, pp. 1770–1777, 2009.

[7] S. Hemminki, P. Nurmi, and S. Tarkoma, "Accelerometer-based transportation mode detection on smartphones," in *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*, p. 13, ACM, 2013.

[8] C. Gershenson, "Artificial neural networks for beginners," *arXiv preprint cs/0308031*, 2003.

[9] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, "Understanding transportation modes based on gps data for web applications," *ACM Transactions on the Web (TWEB)*, vol. 4, no. 1, p. 1, 2010.

[10] L. Bedogni, M. Di Felice, and L. Bononi, "By train or by car? detecting the user's motion type through smartphone sensors data," in *Wireless Days (WD), 2012 IFIP*, pp. 1–6, IEEE, 2012.

[11] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 338–345, Morgan Kaufmann Publishers Inc., 1995.

[14] B. Nham, K. Siangliulue, and S. Yeung, "Predicting mode of transport from iphone accelerometer data," *Machine Learning Final Projects. Stanford University, California*, 2008.

[15] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 10, pp. 1619–1630, 2006.

[16] A. Developer, "Core motion framework reference," 2014.

[17] arturdev, "Somotiondetector." GitHub repository, 2016.

[18] A. Allan, *Geolocation in IOS.* " O'Reilly Media, Inc.", 2012.

[19] D. Mark, J. Nutting, and J. LaMarche, "Where am i? finding your way with core location," in *Beginning iPhone 4 Development*, pp. 557–568, Springer, 2011.

[20] C. Liu, Q. Zhu, K. A. Holroyd, and E. K. Seng, "Status and trends of mobile-health applications for ios devices: A developer's perspective," *Journal of Systems and Software*, vol. 84, no. 11, pp. 2022–2033, 2011.

[21] D. Sambasivan, N. John, S. Udayakumar, and R. Gupta, "Generic framework for mobile application development," in *Internet (AH-ICI), 2011 Second Asian Himalayas International Conference on*, pp. 1–5, IEEE, 2011.