Time series prediction with Fitbit data

Yang Zhang (11529139)

I. Abstract

In this project, I was provided Fitbit time series data from 8 participants. The ultimate goal is to develop an algorithm or a model with adapting existing methods that can predict participants' heathy goal. One of the important step is to build a model that can detect participants' significant changes of their activates and predict their future potential activity pattern. This report mainly focuses on the step of building time series prediction model, the result and analysis is showed in the later part.

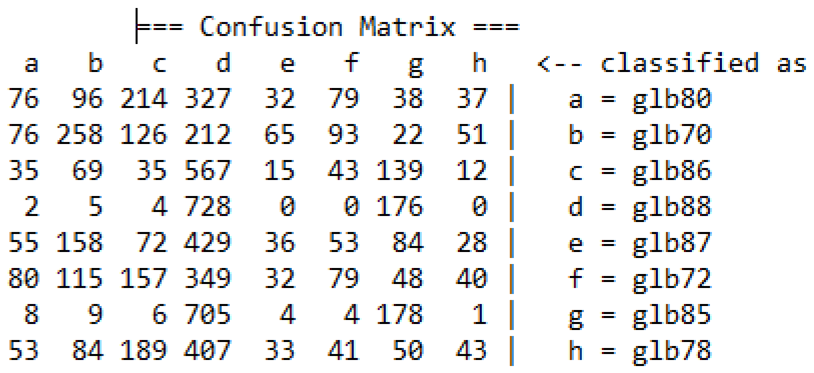
II. Introduction

Based on the different living styles and environments, some people are threatened by potential health risks caused by those factors. Without effectively monitoring their health status, those potential risks could be developed into real health issue. Therefore, it is important that people have their health goals, such as running 5 miles per day or losing 10 pounds' body weight by a single month. Making the goal isn't hard, however, what is hard for people is to make sure they can finish their goal. For steps goal, it is trivial, but for some more complicated health goal it is not obvious. Having a model that can predict people's future activity is significantly helpful, because based on the prediction, people would have a clear look and make changes to their future plans. This is the main motivation of this project.

III. Problem Setup and Related work

The hardware I used is Fitbit tracker and I recurred 8 participants. For the data preparation, I fetched two months of 15-mins interval times series time data (step number) of our participants. I also choose the daily time range from 6 am to 9 pm (15 hours in total), so that I can have less sparse data by excluding most of the inactive hours (such as participants' sleep hours). Then I converted the data into 1 hour interval format for better consistency.

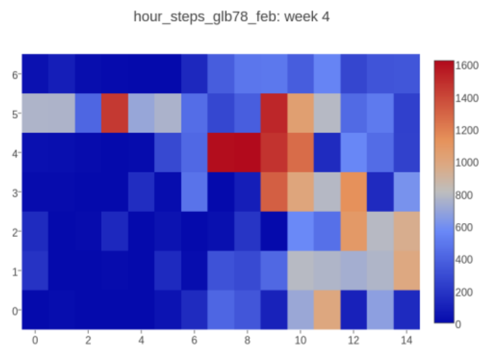
To reach the goal of building a perdition model, I have applied serval machine learning technics. Firstly, I used Weka (an open source machine learning library from University of Waikato) to train the classifier and regression model (each data entry was extract from raw data with format of 3 adjacent hours). The table below is the confusion matrix of the model.



```
            === Confusion Matrix ===
  a   b   c   d   e   f   g   h   <-- classified as
 76  96 214 327  32  79  38  37 |   a = glb80
 76 258 126 212  65  93  22  51 |   b = glb70
 35  69  35 567  15  43 139  12 |   c = glb86
  2   5   4 728   0   0 176   0 |   d = glb88
 55 158  72 429  36  53  84  28 |   e = glb87
 80 115 157 349  32  79  48  40 |   f = glb72
  8   9   6 705   4   4 178   1 |   g = glb85
 53  84 189 407  33  41  50  43 |   h = glb78
```

Unfortunately, Weka didn't generate approving result. The accuracy is only around 20%, and from the confusion matrix, I can see that the model is more likely to classify data with label glb88. The reason is that the glb88 has most general activity pattern, and also because either the algorithm doesn't fit very well or data feature is too sample, it fails to extract information from the data.

Then, I used image processing method with heatmap that generated with participants' weekly hour steps data. The following sample heatmap is from 4th week of Feb. from glb78 (The x-axis is the time and y-axis is the day. Cells with warmer color have high step intensity, vice versa).



I used Keras (an open source neutral networks API) to train the image classifier (32 training samples). After 100 training iteration, the classifier could correctly classify 75% of the testing heatmaps (32 samples)

IV. Solution Approach and Experiments

Solution Approach

I get some result from the methods I mentioned in related work section, but I need the model with more precise and detailed prediction. Therefore, I move my scope to the area of recurrent networks. One of the RNNs called LSTM (Long Short-term Memory) is capable of recording dependencies with large time gap, which is suitable with Fitbit time series data prediction.

LSTM was proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber. The idea of LSTM is that there are three control gate in each of the neutral cell. Those gates are control gate, forget gate and output gate. In high level aspect, the function of forget gate is to decide if the current cell should keep the cell state signal from previous cell. The input gate decides how much new information from feature vector the cell should keep. Lastly, the output gate would generate filtered cell state information and output value based on the control signals from both forget gate and input get.

Experiments

1. Code Preparation

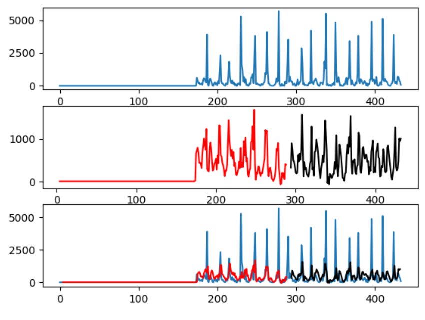
The training and testing environment is built with Python 2.7, and LTSM implementation is imported from Keras API.

2. Data Preparation and Experiment Procedure

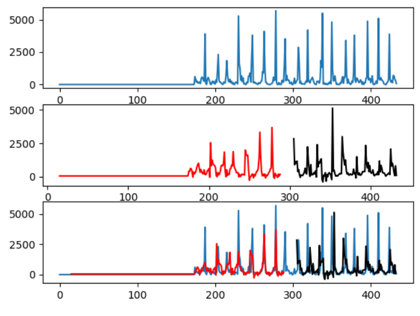
The Raw data of 8 participants was reformatted into (**feature, label**) pair, where the feature is the list of steps numbers that happened before current hour and the label is the step number of current hour. (120 datasets, 15 feature variation for each of the 8 participants). For each datasets, using the first 2/3 portion as training data the rest 1/3 as testing data.

3. Experiment Results

Part of the plots from glb72:



Feature size 1
Training accuracy 0.91
Testing accuracy 0.78



Feature size 4
Training accuracy 0.96
Testing accuracy 0.94

Each of the plots is composed with 3 subgraphs, the top subgraph is the original step number plot (ground truth plot). The middle graph shows the predicted step numbers (red curve is for training prediction and the black curve is for testing prediction). The 3rd subgraph is the super-composition plot with the previous two subgraphs. From the plots, we can see that even though the LSTM model trained with size 1 feature data can predict the significant changes well, it cannot predict the peak value precisely. However, when change the feature size to 4, the peak prediction is significantly improved. This tells us different feature size may affect the performance. The following table shows the tuned feature size of each participant

(Note the accuracy is calculated from equation 1- abs(prediction - truth)/truth).

Participant	Best Feature size	Training Accuracy	Testing Accuracy
glb70_feb	3	0.9752	0.9610
glb72_feb	4	0.9635	0.9414
glb78_feb	1	0.9533	0.9086
glb87_june	2	0.9772	0..9332
glb89_june	3	0.9794	0.9354
glb91_june	4	0.9660	0.9334

V. Conclusions

LSTM works really well in this experiment setting. With the dataset of tuned feature size, LSTM can reach 90% accuracy on both training data and testing data. However, there is a limitation of my experiment. For the testing data, I assume that the features are correct which means the feature data is constructed from ground truth. This is not a problem when only predict the next one hour, however, if use the model to predict next a few hours, the accuracy may be off from the true accuracy.

As a conclusion, this project provides me hand-on the experience of handling time series data. Moreover, along with the project going, I learnt variable skills not only related to algorithms but also the skills that transferring machine learning knowledge into real insights to end-users. More importantly, it will guide me in the future in the road of pervasive computing area.

VI. Acknowledgments and References

1. Sepp Hochreiter and Jürgen Schmidhuber, Long Short-Term Memory. Neural Computation 9(8):1735-1780, 1997
2. Keras is an open source library https://github.com/fchollet/keras