Task A: Multimodal activity recognition: Modes of locomotion

Aamena Alshamsi -PhD Student (Computing and Information Science Program)

Masdar Institute of Science and Technology, Abu Dhabi, UAE

Abstract

The goal of this task is to classify modes of locomotion from body-worn sensors. Hence, I propose a solution including three steps involved in activity recognition process. First, I select the features to be included in the recognition process. Second, I reduce the dimensionality of the given datasets by transforming them using Principal Component Analysis (PCA). Third, I use K Nearest Neighbor algorithm to recognize the locomotion.

Feature Selection, Dimensionality Reduction and Feature Transformation

All the features are considered for activity recognition except the first time feature which is measured in milliseconds because of its low importance in comparison to other features.

Due to the large number of provided features, a dimensionality reduction method is used. Principal Component Analysis (PCA)(Pearson, 1901) is used to map the high dimensional dataset into a smaller one in order to remove irrelevant features and remove correlation among some other features. I use Weka(Weka 3: Data Mining Software in Java)to perform PCA on the datasets with a Ranker search which selects the highly ranked features due to “the amount of variance each accounts for”(Principal Component Analysis with Experimenter, 2011).The chosen percentage of the variance in the original data is default 0.95 (95%). Transformed dataset is obtained individually for each test subject datasets. Thus, one PCA transformation dataset for Subject 2 including all the subjects’ s datasets (S2-ADL1,S2-ADL2,S2-ADL3,S2-ADL4,S2-ADL5,S2-ADL-Drill) and another PCA transformation dataset for Subject 3 including all the subject’s datasets (S3ADL1,S3-ADL2,S3-ADL3,S3-ADL4,S3-ADL5,S3-ADL-Drill).The selected features by PCA for Subject 2 are 69. While the selected features for Subject 3 are: 68

Recognition Algorithm

I use K nearest neighbor KNN to (Alpaydın, 2004) recognize the classes of test datasets. I ran the algorithm on transformed datasets (as explained in the preceding section). The setting of the algorithm is default in Weka (IB 1). Normalized Euclidean distance is used to find the closest training point to the testing point. Again, the algorithm is run on each transformed dataset separately. Thus, training is subject-dependent.

Bibliography

(n.d.). Retrieved September 21, 2011, from Weka 3: Data Mining Software in Java: http://www.cs.waikato.ac.nz/ml/weka/

Principal Component Analysis with Experimenter. (2011, August 3). Retrieved September 21, 2011, from http://old.nabble.com/: http://old.nabble.com/Principal-Component-Analysis-with-Experimenter-td32186953.html

Alpaydın, E. (2004). *Introduction to Machine Learning.* MIT Press.

Pearson, K. (1901). On Lines and Planes of Closest Fit to Systems of Points in Space. *Philosophical Magazine 2* , pp. 559–572.

Opportunity Activity Recognition Challenge Submission Report

Zahraa Said Abdallah
Centre for Distributed Systems and Software Engineering
Monash University
900 Dandenong Rd, Caulfield East, VIC3145
Australia
Email:zahraa.said.abdallah@monash.edu

***Abstract*—This report explains the implementation details for activity recognition challenge submission. Description of the training model, features selected, and proposed method is represented in this report.**

I. Introduction

The submitted results are for Task A to classify the locomotion of unlabelled data in both subjects 2 and 3 with j48graff algorithm.

II. Training model

Activity recognition algorithms are typically using models created by the classification methods. Training labelled data is been used to create a model for the different activities. These data serve as examples to the classification method, so it can associate certain attributes of the data with each activity. Each training data point includes the label of its associated activity as well as the attributes that are extracted as indicators of the activity. When the model is ready, it can be used to classify unlabelled data.

The accuracy of the classifier strongly relies on the robustness of the training model. Therefore, Over-fitted or poor models could be the main reason for poor classification performance.The possibility of over-fitting exists when the criterion used for training the model is not the same as the criterion used to judge the efficacy of a model. In particular, a model is typically trained by maximising its performance on some set of training data. However, its efficacy is determined not by its performance on the training data but by its ability to perform well on unseen data.

The data used for the challenge is composed of the recordings of four subjects. Two types of recording sessions were performed: Drill sessions where the subject performs sequentially a pre-defined set of activities and "daily living activities" runs (ADL) where he executes a high level task (wake up, groom, prepare breakfast, clean) with more freedom about the sequence of individual atomic activities[1]. Therefore, building the training model from daily living activities should achieve a better performance when testing on data with the same criterion. For each subject, the training model is built from the labelled "daily living activities" distributed in the three segments(ADL1, ADL2, ADL3).

All unrecognised activities in the training examples have been omitted and ignored in building the training model to enhance the classification performance.

III. Features selection

Challenge dataset composites of 114 features including 36 features from the accelerometer sensors and 77 from the inertia sensors and one feature representing time.

The only feature that has been ignored for both training and testing was the time feature. As The time feature is representing the time elapsed since the start of recording , it has no effect on the type of classified locomotion for daily live activities.

IV. Algorithm

Methods deployed for activity classification were recently reviewed in [2]. Methods like artificial neural networks, support vector machines, K-nearest neighbour, decision trees, Bayesian classifiers, etc. are commonly used. However, decision tree has been popularly deployed for activity recognition because of its simplicity and efficiency.

Decision tree grafting has been deployed for classifying the challenge data. Decision tree grafting adds nodes to an existing decision tree with the objective of reducing prediction error. Grafting new nodes to correct poor classification can significantly improve the predictive accuracy of the inferred decision trees.The details of the grafted decision tree is found in [3].

J48graft Weka implementation [4] has been used for training and testing purposes.

A. Parameters

The parameters used for algorithm implementation with their set values are described as follow:

- binarySplits: Whether to use binary splits on nominal attributes when building the trees. The value has been set to: *False*.
- confidenceFactor: The confidence factor used for pruning The value has been set to 0.25.
- minNumObj: The minimum number of instances per leaf. The value has been set to 2.

- subtreeRaising: Whether to consider the subtree raising operation when pruning.The value has been set to *True*.
- unpruned: Whether pruning is performed. The value has been set to *False*.
- useLaplace: Whether counts at leaves are smoothed based on Laplace .The value has been set to *False*.

REFERENCES

[1] Challenge opportunity website: http://www.opportunity-project.eu/challengeDataset

[2] S. J. Preece, J. Y. Goulermas, L. P. J. Kenney, D. Howard, K. Meijer, and R. Crompton, : Activity identification using body-mounted sensors-A review of classification techniques, Physiological Measurement, vol. 30, pp. R1-R33,2009.

[3] G. I. Webb: Decision Tree Grafting From the All Tests But One Partition. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI '99), Thomas Dean (Ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 702-707, 1999.

[4] I. H. Witten and E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed.San Francisco, CA: Morgan Kaufmann,2005.

Description of Our Proposed Method (Task A, B1, B2)

Participants:

Hong CAO

Minh Nhut NGUYEN

Xiao-Li LI

Shonali Priyadarsini KRISHNASWAMY

1. Overview

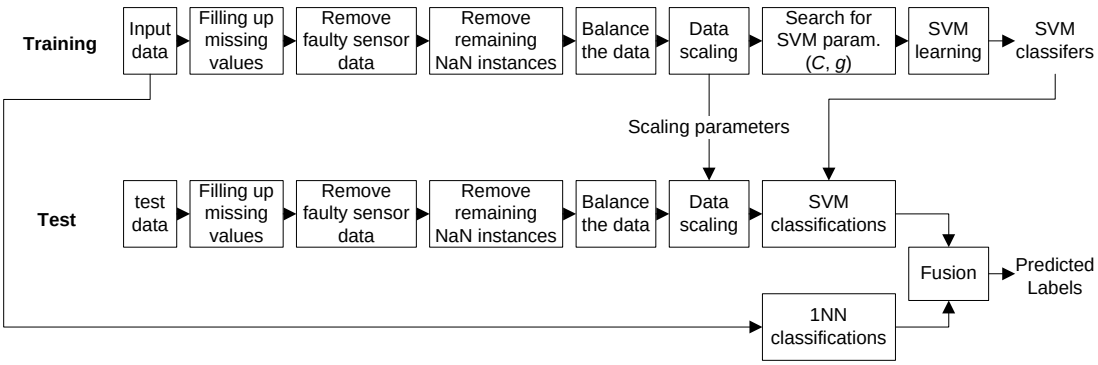


Fig. 1 Block diagram of our methods

Fig. 1 shows the block diagram of our proposed method. This involves a number of data processing techniques such as filling missing data, removing faulty sensor data, removing remaining NaN instances, balancing the data, scaling as well as SVM learning. In the testing phase, we fuse our SVM classification results and the 1NN results. Also by considering the sequence nature, we smooth out our classification output.

2. Data Preprocessing

Filling up the Missing Data

We use spline interpolation together with the time data to perform interpolation using the Matlab function *interpl*. An example code block is shown below.

…

```
b = Y(:,1); %Time axis
a = Y(:,j); %A data col with missing data

idx = find(~isnan(a));

if length(idx) < m && length(idx) > 1 %fill the missing data
    a = interp1(b(idx),a(idx),b, 'v5cubic');
end
...
```

Remove the Fauly Sensor Data

We find that for several sensors (Column 35-37), the majority of their data are missing. As it is hard to give them good approximation, we simply remove these data. Also we remove the columns corresponding to the time (first column) and the gesture label (the last column). For gesture recognition, we remove the locomotion labels (the second last columns).

Remove the NaN Rows

After the above two steps, we find there are always records at the bottom of each data file still containing NaNs. This is because we choose to perform intepolation only (no extrapolation) in filling up the missing data. We find all these remaining NaN records correpond to Null records. Therefore, we simply remove them from our training and testings.

Balance the Data

We find that for the classification tasks, the training data from the different classes are fairly imbalanced. Note that in solving tasks A, B1 and B2, we used all the data in drill, adl1, adl2 and adl3 for training after removing the Null records. For Task A, as data from the four locomotions are fairly imbalanced, we use oversampling and undersampling technique to prepare 20,000 records per class for learning. For Task B1, we select all activity records and randomly choose the same total number of records from the null class. For Task B2, we use undersampling and oversampling to have 2000 records for each class.

For classes with more than our required learning records, we perform random undersampling to select the needed records.

For classes with less than our required learning records, we perform structure preserving oversampling (SPO) to generate synthetic samples to make up for the additional records needed. The SPO algorithm is aimed to create synthetic samples by preserving the current covariance structure and intelligently generating some protective variances in the trivial eigen dimension. One can find our paper in https://sites.google.com/site/sstarcao/home .

Data Scaling

We find the maximal and the minimal values for each feature. Then, we perform the linear feature scaling to normalize each feature into the range [-1, +1].

Searching for Best SVM parameters

Based balanced and scaled training feature set, we perform grid searching for the best parameters (*C*, *g*) in log scale, which give the maximal recognition accuracy. The parameters are associated with SVM learning with radial basis kernel. We find that our best (C, g)s found are either (32, 0.5) or (8, 0.5) for the different tasks with S2 and S3.

To improve the speed, this process sometimes is performed on a random down-sampled dataset.

3. SVM Learning and Classification

We perform SVM training with the best parameters to learn a four-class classifier (Task A), 2-class classifier (Task B1) and 18-class classifier (Task B2) using LIBSVM tool. The classifier is then used to predict the labels for the testing data. After filling the missing data and removing the faulty sensor data, if a record still contains NaN, we simply classify the record to the Null record without using our SVM classifier.

For the 1NN classification, we get the classification outcomes based on all the training data and the scaled features.

Our fusion of the two decision outcomes is through interleaving the SVM labels and the 1NN labels and then performing median filter-alike smoothing operations. Our Matlab code for this part is attached in this appendix (also as email attachments)

Appendix

```
%This is the code for integrating the classifications from SVM and 1NN
for
%Task A, Subject 2 and ADL4

%Including paths
addpath('C:\Users\Cao Hong\Desktop\Challenge Act
Recognition\Submission\SSTAR_Hong');
addpath('C:\Users\Cao Hong\Desktop\Challenge Act
Recognition\Submission\NSTAR_Minh');

Filt_Len = 9; %for Task A. For task B1 and B2, we choose Filt_Len = 61


%Load Data
load A_2_SSTAR.mat adl4
load A_2_NStar.mat A_S2_ADL4

S = adl4;        %SVM Labels
K = A_S2_ADL4;   %KNN Labels

if size(S, 1) ~= size(K, 1)
    error('Mismatching label dimensions');
end

len = length(S); % the length

%Creating Combined interleaved labels
J = zeros(2*len,1);
for i = 1:len
    J( i*2-1, 1 ) = S(i,1);
    J( i*2, 1)    = K(i,1);
end

D = medfilt(J, Filt_Len);
J = D;

%Flag = 1 if a confusion is found
Flag = ones(len,1);
C    = zeros(len,1); % To store the combined end labels
for i =1:len
    if J(2*i-1,1) == J(2*i,1)
        C(i,1) = J(2*i);
        Flag(i,1) = 0;
    end
end
```

```
%Iteratively resolve the confusions
Ite = 0;
while sum(Flag) > 0

    Ite = Ite + 1;
    idx = find(Flag==1);

    if Ite >=50
        for i = 1:length(idx)

            C(idx(i),1) = J(2*idx(i)-1,1);
            Flag(idx(i)) = 0;
        end
        break;
    end

    for i = 1:length(idx)

        pos = idx(i);

        sAgr = 0;
        kAgr = 0;

        if pos - 1 > 0
            if Flag(pos-1) == 0
                if J(2*i-1,1) == C(pos-1,1)
                    sAgr = sAgr + 1;
                end

                if J(2*i,1) == C(pos-1,1)
                    kAgr = kAgr + 1;
                end
            end
        end

        if pos + 1 < len
            if Flag(pos+1,1) == 0
                if J(2*i-1,1) == C(pos+1,1)
                    sAgr = sAgr + 1;
                end

                if J(2*i,1) == C(pos+1,1)
                    kAgr = kAgr + 1;
                end
            end
        end

        if max([sAgr kAgr]) >0
            if sAgr >= kAgr
                C(pos,1) = J(2*pos-1,1);
            else
                C(pos,1) = J(2*pos,1);
            end
            Flag(pos) = 0;
        end
```

```
    end

end

adl4=C;

save A_2_4_CSTAR adl4

%EOF

%-------------------------------------------------------


%This is the function for Median filter-alike smoothing
function D = medfilt(J, Filt_Len)

len = length(J);
D= J;

Thr = 0.4*Filt_Len;

for i= ((Filt_Len+1)/2) : (len - (Filt_Len-1)/2)

    neighbor = J( (i-(Filt_Len-1)/2) : (i+(Filt_Len-1)/2), 1 );
    idx = find( neighbor~=0);

    if length(idx) > Thr
        D(i,1) = mode(neighbor(idx,1));
    else
        D(i,1) = 0;
    end

end

%EOF
%-------------------------------------------------------------
```

OPPORTUNITY CHALLENGE

TEAM: WASNLab Team
PARTICIPANTS: Matteo Giuberti
AFFILIATION: University of Parma, Italy

General Description of the Approach used for the contest

TASK A

For the detection of the locomotion states, some considerations are made for every locomotion and some constraints are to be fulfilled by the data in order to consider that the user is doing a specific locomotion.
The different subjects have been treated independently.
Many windows are computed from the test datasets (ADL 4 and 5), for every locomotion.
Each of the previous window satisfies the constraints of the relative locomotion and every locomotion constraint can include different subsets of the sensors used by the user.
There are some priorities between the different locomotions: “lie” has the major priority, followed in order by “sit”, “walk”, and “stand”.
If more than one locomotion window overlap on the same samples the window with the highest priority is taken and the others are deleted.
Some constraints are considered for “lie”, “sit”, and “walk” whereas no constraints are used for “stand”. This because “stand” is always detected when no other locomotion states are detected.

TASK B1-B2-C

The same approach has been used for task B1, B2, and C, and the different subjects have been treated independently.
The approach consists of two phases: a training phase and a testing phase.

During the training phase, the Drill set has been fused together with ADL 1, 2, and 3 sets. For each occurrence of every gesture, isolated on the basis of the provided labels, different features has been collected: the mean, the standard deviation, the minimum and maximum values, and the duration (in samples).
Then, for every gesture the following values has been stored: the maximum value of all the maximum values (V), the minimum value of all the minimum values (v), the maximum and the minimum values of the mean (M and m), the maximum value of the standard deviation (S), the minimum value of the duration (d).
After that, a testing phase has been applied to the data of the training phase in order to observe the performance of the system on the same data used for the training phase.
A testing phase consists in the following operations: for every gesture a scan of every sample of the dataset used is made and if the sample, that we call x, satisfy the following constraint:
$v<x<V$

this sample is marked as a possible occurrence of the relative gesture.
Completed the scan we will have many windows of samples in which the gesture could have been verified.
The window that has duration lower than d are then discarded.
Finally, only the windows with a mean c that satisfy the following constraint are considered:
$m-S<c<M+S$
For each gesture, the resulted labels (computed from the previous windows) are then compared with the actual labels and the performance of the detection of the gesture is stored.

During the testing phase, instead, the test dataset (ADL 4 and 5) are taken and the same process described before is done on the data, except the computation of the performance.
The labels of every gesture are then fused together and, whenever more than one gesture is detected during a specific sample, the gesture with the best performance (computed during the training phase) is chosen.

For task B1, just “0”s and “1”s are considered and the gestures are not classified.

Opportunity Challenge Submission for Task C

Naveen Nair[123], Amrita Saha[2], Ganesh Ramakrishnan[21], and Shonali Krishnaswamy[31]

[1] IITB-Monash Research Academy, Old CSE Building, IIT Bombay
[2] Department of Computer Science and Engineering, IIT Bombay
[3] Faculty of Information Technology, Monash University
{naveennair,amrita,ganesh}@cse.iitb.ac.in
Shonali.Krishnaswamy@infotech.monash.edu.au

Abstract. Hidden Markov Model (HMM), due to its ability to capture temporal dependencies along with the emission dependencies, has been an efficient tool for activity recognition systems [1][2]. In this submission, we model the given problem using HMM for activity recognition and report the results obtained on Task C data.

1 Team Name: NAGS

2 Description of the method used

In activities of daily living, the activity at a particular time step depends on the activity that happened in the previous time step. Also the activity that is happening at a particular time instance triggers some sensors whose values are observable. Since Hidden Markov Model (HMM) is a probabilistic model that captures the above two types of dependencies, we formulate our problem in an HMM setup. We give a brief description of HMM for activity recognition in the next paragraph.

In an activity recognition setting, the joint state of sensor values recorded for every time unit forms the observation/emission in a Hidden Markov Model (HMM) and is represented as $\mathbf{x}_t$ at time t. The activities for each time unit is the hidden state/label, represented as y_t at time t, of HMM as shown in Figure 1. In a typical HMM setup, y_t at time t is independent of all other variables given y_{t-1} at time $t-1$ and $\mathbf{x}_t$ at time t is independent of all other variables given y_t [1][2]. Given the independece assumptions, the joint distribution of the sequence of labels (Y) and observations (X) of length l can be expressed as $P(X,Y) = \prod_{t=1}^{l} P(y_t|y_{t-1})P(\mathbf{x}_t|y_t)$, where $P(y_1|y_0)$, $P(y_t|y_{t-1})$, and $P(\mathbf{x}_t|y_t)$ stands for the initial state distribution, the transition distribution, and the emission distribution respectively [4][3]. During the training phase, parameters are learned by maximizing the joint probability, $P(X,Y)$, in the training data. For prediction, the learned parameters are used to determine the label (activity) sequence that best expains the observation (joint state of sensors) sequence. A dynamic programming algorithm called the Viterbi Algorithm [5] is used to find

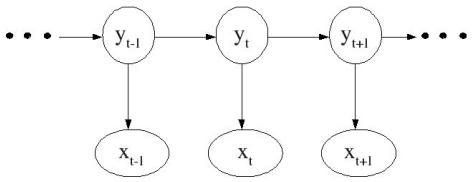


Fig. 1: Graphical representation of an HMM. Circles represent hidden states and ellipses represent observable variables

the maximum probable label sequence. We now discuss the experiments and results in the next section. We now discuss the experiments and results in the next section.

3 Experiments and Results

For our implementation, we discretized the input parameters for Task C data and ran Hidden Markov Model on that. For naive factorization, a conditional independence is assumed among the observations (sensor values) given an activity. All the five training examples are used for training. Inference is done by dynamic programming algorithm, Viterbi [5]. Inference is done on S4-ADL4-noisy.dat and S4-ADL5-noisy.dat data. The results are attached. The result file names are "C_4_ADL4_NAGS.txt" and "C_4_ADL5_NAGS.txt". These are results of input files "S4-ADL4-noisy.dat" and "S4-ADL5-noisy.dat" respectively.

4 Conclusion and Future Work

We have modeled the give problem for Task C using HMM for activity recognition. We have run experiments for Task C and the results are attached.

References

1. Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne and Ben krose, *Accurate activity recognition in a home setting*, 10th International conference on Ubiquitous computing, 2008.
2. C.H.S. Gibson, T.L.M. van Kasteren and Ben Krose, *Monitoring Homes with Wireless Sensor Networks*, Proceedings of the International Med-e-Tel Conference, 2008.
3. R. Rabiner, *A tutorial on hidden Markov models and selected applications in speech recognition*, Proceedings of the IEEE, 77(2):257–286, 1989.
4. Lise Getoor and Ben Taskar, *Statistical Relational Learning*, MIT Press, 2006.
5. Forney GD, *The viterbi algorithm*, Proceedings of IEEE, 61(3):268–278, 1973

Description of Our Proposed Method (Task A, B1, B2)

Participants:

Minh Nhut NGUYEN

Hong CAO

Xiao-Li LI

Shonali Priyadarsini KRISHNASWAMY

I. Data Preprocessing

1. Filling up the Missing Data

We use spline interpolation together with the time data to perform interpolation using the Matlab function *interpl*. An example code block is shown below:

```
…
b = Y(:,1); %Time axis
a = Y(:,j); %A data col with missing data
idx = find(~isnan(a));
if length(idx) < m && length(idx) > 1 %fill the missing data
   a = interp1(b(idx),a(idx),b, 'v5cubic');
end
…
```

2. Remove the Fauly Sensor Data

We find that for several sensors (Column 35-37), the majority of their data are missing. As it is hard give them a good approximation, we simply remove these data. Also, we remove the columns corresponding to the time (first column) and the gesture label (the last column).

3. Remove the NaN Rows

After the above two steps, we find that there are always records at the bottom of each data file still containing NaNs. These remaining NaN records correspond to the Null records. Therefore, we simply remove them from our training and testing data sets.

4. Data Scaling

We use the z-*scores* normalization on each sensor's signals.

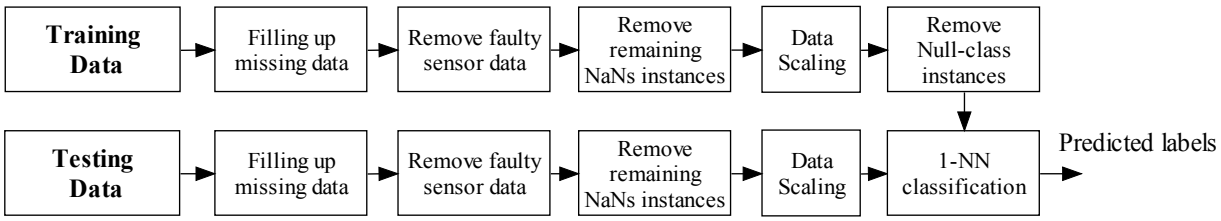
5. Remove Null-class instances

We remove all the Null-class instances in the training data set for task A.

II. Classifier and models

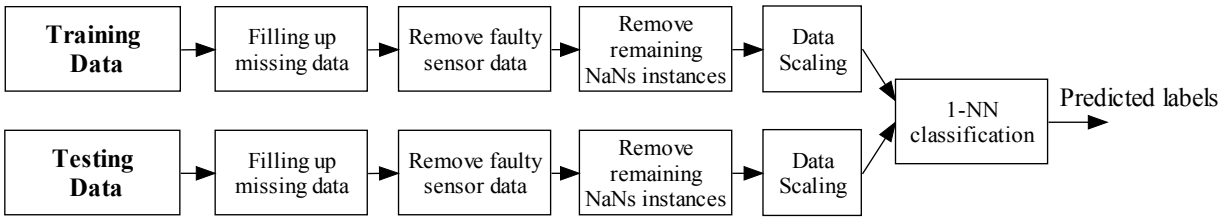
Although there have been many algorithms proposed for time series classification, interestingly, the simple technique 1-NN classification based on the top one nearest neighbor with Euclidean distance was shown to be the very competitive technique. Therefore, we use the 1-NN classification for all the tasks.

Task A: Multimodal activity recognition: Modes of locomotion



The label files are named as: A_2_NStar.mat and A_3_NStar.mat.

Task B: Automatic segmentation and Multimodal activity recognition: Gestures



We use the same mode for both task B1 and task B2. However, in task B1, the predicted values for all the activity classes are set to **1**.

The label files are named as: B1_2_NStar.mat, B1_3_NStar.mat, and B2_2_NStar.mat, B2_3_NStar.mat.

Description of Our Proposed Method (Task A, B1, B2)

Participants:

Hong CAO

Minh Nhut NGUYEN

Xiao-Li LI

Shonali Priyadarsini KRISHNASWAMY

1. Overview

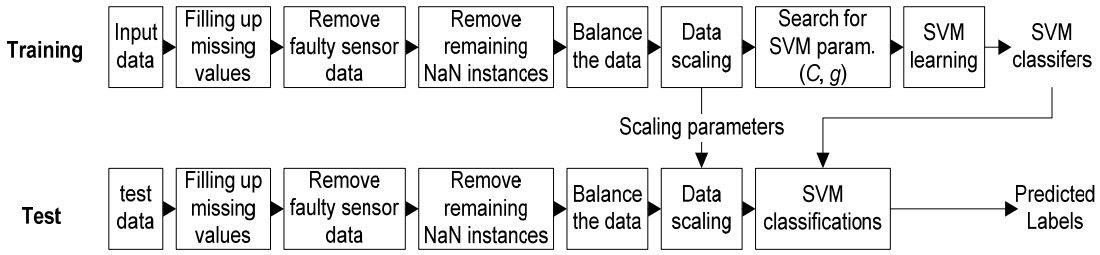


Fig. 1 Block diagram of our methods

Fig. 1 shows the block diagram of our proposed method. This involves a number of data processing techniques such as filling missing data, removing faulty sensor data, removing remaining NaN instances, balancing the data, scaling as well as SVM learning.

2. Data Preprocessing

Filling up the Missing Data

We use spline interpolation together with the time data to perform interpolation using the Matlab function *interpl*. An example code block is shown below.

```
...
b = Y(:,1); %Time axis
a = Y(:,j); %A data col with missing data

idx = find(~isnan(a));

if length(idx) < m && length(idx) > 1 %fill the missing data
```

```
    a = interp1(b(idx),a(idx),b, 'v5cubic');
end
...
```

Remove the Fauly Sensor Data

We find that for several sensors (Column 35-37), the majority of their data are missing. As it is hard give them good approximation, we simply remove these data. Also we remove the columns corresponding to the time (first column) and the gesture label (the last column).

Remove the NaN Rows

After the above two steps, we find there are always records at the bottom of each data file still containing NaNs. This is because we choose to perform intepolation only (no extrapolation) in filling up the missing data. We found all these remaining NaN records correpond to Null records. Therefore, we simply remove them from our training and testings.

Balance the Data

We find that for the classification tasks, the training data from the different classes are fairly imbalanced. Note that in solving tasks A, B1 and B2, we used all the data in drill, adl1, adl2 and adl3 for training after removing the Null records. For Task A, as data from the four locomotions are fairly imbalanced, we use oversampling and undersampling technique to prepare 20,000 records per class for learning. For Task B1, we select all activity records and randomly choose the same total number of records from the null class. For Task B2, we use undersampling and oversampling to have 2000 records for each class.

For classes with more than our required learning records, we perform random undersampling to select the needed records.

For classes with less than our required learning records, we perform structure preserving oversampling (SPO) to generate synthetic samples to make up for the additional records needed. The SPO algorithm is aimed to create synthetic samples by preserving the current covariance structure and intelligently generating some protective variances in the trivial eigen dimension. One can find our paper in https://sites.google.com/site/sstarcao/home .

Data Scaling

We find the maximal and the minimal values for each feature. Then, we perform the linear feature scaling to normalize each feature into the range [-1, +1].

Searching for Best SVM parameters

Based balanced and scaled training feature set, we perform grid searching for the best parameters (*C*, *g*) in log scale, which give the maximal recognition accuracy. The parameters are associated with SVM learning with radial basis kernel. We find that our best (C, g)s found are either (32, 0.5) (8, 0.5) for the different tasks with S2 and S3.

To improve the speed, this process sometimes is performed on a random down-sampled dataset.

3. SVM Learning and Classification

We perform SVM training with the best parameters to learn a four-class classifier using LIBSVM tool (select 20,000). The classifier is then used to predict the labels for the testing data. After filling the missing data and removing the faulty sensor data, if a record still contains NaN, we simply classify the record to the Null record without using our SVM classifier.

Activity Recognition Challenge Task A: Recognizing Locomotions by Dimention Reduction and Boosting

Shoji Tominaga, Masamichi Shimosaka, Rui Fukui, and Tomomasa Sato
Intelligent Cooperative Systems Laboratory
Dept. of Mechano-Informatics, The Univ. of Tokyo
Tokyo, Japan
{tominaga,simosaka,fukui,tsato}@ics.t.u-tokyo.ac.jp

***Index Terms*—activity recognition, wearable sensors, boosting, principal component analysis**

I. Introduction

To achieve accurate recognition from data of wearable sensors, a data-oriented approach is deployed. We assume that the sensor data themselves have sufficient information to classify the locomotion classes. For example, the direction of gravity are obviously different between lying and standing. However, it has the risk of overfitting to straightly utilize the naive classifiers e.g. decision stump. In addition, some of the test data have lack of values. We solve these problems simply but certainly, by selecting sensors and learning classifiers using principal component analysis (PCA) and boosting.

II. Methods

A. Procedures

Our method proceeds with the following:

1) Extracting feature vectors v
2) Compressing v into v_c by PCA
3) Learning one-versus-the rest classifiers of each class of locomotion by adaboost
4) Classifying the test data by all 4 classifiers
5) For each frame of test data, decide the class by majority rule of neighborhood

Followings are the details of each procedure.

1) Since there are lack of values in both teacher and test data, We eliminate sensors which have many lacks. We use sensor values themselves, mean and variance of neighborhood as features of each sensor. That is, the dimension of original feature vectors v is three times of the numbers of sensors.

2) Before the compression, each feature are normalized into the same mean and variance. Then v is compressed by:

$$v_c = (c_1 ... c_n)^{\mathsf{T}} v \tag{1}$$

Where c_i is the i-th principal component of $\{v\}$

3) To learn classifier with adaboost, each weak classifier is the one-dimensional linear discrimination i.e.

$$\phi_{i,j,k}(v_c) = \mathrm{sign}(v_c(i) > j)^k \tag{2}$$

Where i is the used dimension, j is the threshold, and $k = \{-1, 1\}$ is the orientation of the classifier.

4) When the method classifies the test data, features of them are extracted from the same sensor as these used for learning classifier and compressed by the same matrix of step 2.

As a special case of step 4 and 5, if there are frames which have lack of values except those already eliminated in step 1, the method skips to classify them and decide the class of locomotion by that just before them.

B. Settings of Parameters and Other Condition

In this subsection, we note the settings for adapting our method to the dataset[1], [2].

The parameters of our method is,

- The frames of neighborhood to extract features (step 1)
- The number of dimensions of compressed features by PCA (step 2)
- The number of weak classifiers (step 3)
- The frames of neighborhood to decide classes by majority rule (step 5)

We used 30 frames for features (e.g. means of frame 100 were evaluated from frame 70-130), 15 dimensions, 40 weak classifiers, 40 frames for majority rule (it was the same as that for features) from our experience.

As other condition, the list of eliminated sensors are in (table). The numbers in the table correspond to those of the original data (e.g. 5 is the value of axis x of accelerometer on the hip). For reducing calculation time, we used half of the extracted features of standing, walking, and sitting as teacher data. Since the data of lying are much less than that of the others, we used all of them.

TABLE I
Index of Unused Sensors for Learning Classifiers

Test Data	Unused Sensor Index
S2_ADL4	8-10, 14-16, 20-22, 20-31, 35-37
S2_ADL5	2-4, 8-10, 14-16, 35-37
S3_ADL4	2-4, 11-13, 20-22, 35-37
S3_ADL5	2-13, 20-22, 35-37

References

[1] D. Roggen et al., "Collecting complex activity data sets in highly rich networked sensor environments," in *Seventh International Conference on Networked Sensing Systems*, 2010.
[2] P. Lukowicz et al., "Recording a complex, multi modal activity data set for context recognition," 2010.

Opportunity Activity Recognition Challenge using HASC Tool

Tianhui Yang **Nobuhiro Ogawa** **Yohei Iwasaki** **Katsuhiko Kaji** **Nobuo Kawaguchi**
Graduate School of Engineering, Nagoya University
{tenki, hiro, iwasaki, kaji, kawaguti}@ucl.nuee.nagoya-u.ac.jp

ABSTRACT
To accelerate and simplify human activity recognition research, we have been developing a data processing tool named "HASC Tool[1][2]." In this paper, we performed a data processing mechanism used for opportunity dataset which is implemented in the HASC Tool. By using the system, we finished Task A of Opportunity Activity Recognition Challenge. We also show the preliminary experimental result.

Keywords Activity Recognition, Activity Understandings, Opportunity, HASC Tool.

INTRODUCTION

In this paper, we introduce our method to classify locomotion (Task A). By using the part of HASC Tool, we performed a large number of evaluations with the user-dependent data from opportunity dataset.

In the following section, we first explain the toolkit we use to analysis the data. And then report the sensors we use. Third, we will report the method we process the activity dataset. We also report the features we use to learn the data. The last part of our paper will show the result and conclusion of the experiment.

TOOLKIT FOR ACTIVITY RECOGNITION

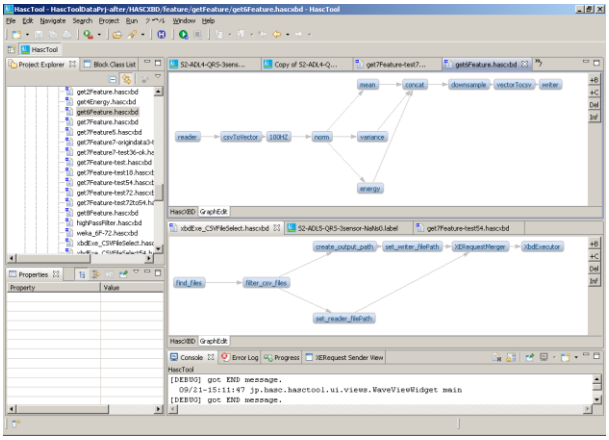


Figure1. Screen shot of the HASC Tool (XBD file)

To boost the data handling and trial-and-error process of the signal processing, we have developed a toolkit named "HASC Tool." Figure 1 and 2 show screen images of HASC Tool. HASC Tool is developed with Java and based on the famous IDE called Eclipse RCP. HASC Tool has following features.

- Showing accelerometer signals and label data (Figure.2)

- Create a process block diagram graph called "XBD." By using "XBD," one can easily automate the various signal processing and file processing (Figure.1). Without this kind of automation, handling thousands of files is not easy.

- Real time / offline data acquisition with wireless sensors

- Connection with WEKA Toolkit

By using HASC Tool, we can exchange the process of activity recognition using XBD files.

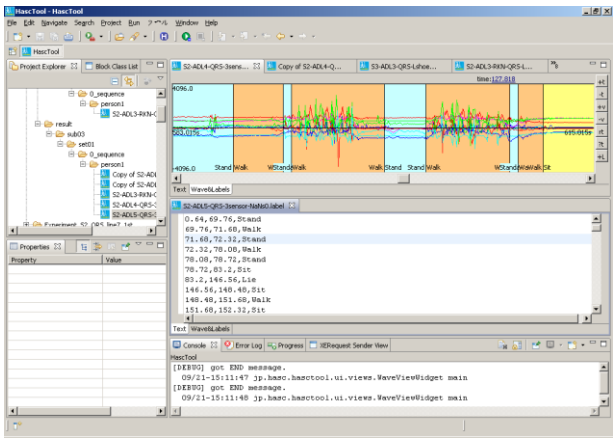


Figure2. Screen shot of the HASC Tool (wave view)

THE SENSORS WE CHOOSE

In the experiment, we use only 3 sensors, RKN^ Accelerometer, BACK Accelerometer and L-Shoe Body Accelerometer. Because opportunity dataset is a database of daily activities recorded in a sensor rich environment. It is very important to choose appropriate sensors. Here, I will give some reasons why I choose these sensors. First, the RKN^ accelerometer is on the upper side of the knee, so it is easy to be used to classify the locomotion of siting and standing. Second, we use L-Shoe Body Accelerometer on the left foot, because this sensor can tell the difference between walking and standing. Third, to classify the activity between lying and siting, we use the accelerometer on the back. It can easily identify the activity of lying.

METHOD

To create the learning data, we segment the Drill data into many files by different labels. There is some "NaN" data in the datasheet, so we do not use the segments which include "NaN" data of any of the 3 sensors as the learning data. There are no data with "lie" label in the Drill data, so we use "lie" data from ADL1 and ADL2 as the learning data. We do not use the data of "lie" from ADL3 because we

want to use ADL3 to test our results. There are many activities last only a very short while and it is difficult to get features from them, so we use the activities longer than about 6 seconds as the learning data. And we also delete some bad learning data to raise our recognition rate. In HASC Tool data format, the labels and data are in different files. So we have converted the data into HASC format, so we can process the dataset in HASC tool. We use our toolkit HASC tool to get features from learning data.

SELECTION OF FEATURE

There are many researches on the field of activity recognition. Bao[6], Chang[4] and Lee[5] conducted the activity recognition by using some features of activity data and applied them to machine learning. They used various features such as mean, variance, standard deviations, energy and correlation features. From the purpose of this experiment to be a basic reference data, we used only simple features which are used by many researchers. In this experiment, activity data is a sequence of 3-axis accelerometer signal. We evaluated the activity data using 7 features of each sensor's axis, the features are mean, variance, and energy of each frequency band (four types). So the totally we use 54 features used for activity recognition. In the experiment, features were computed mean and variance features on 256 samples windows of acceleration data and the samples of windows on energy are 128. The samples overlapping between consecutive windows are 56. We used C4.5 decision tree [3] on WEKA toolkit. We use the user-dependent data analysis to conduct the activity recognition.

RESULT OF TASK A

To test the learning algorithm and the learning dataset of S2 and S3, we use S2-ADL3 and S3-ADL3 to test the recognition rate. The result of S3-ADL3 is shown in Table 3. From the result of experiment on S2-ADL3, we got a total recognition rate of 88.56% and the result from S3-ADL3 is 88.02%. More details about the results of S3-ADL3 are shown in the Confusion Matrix of table 1 and 2.

CONCLUSION AND FUTURE WORKS

In this paper, we report the human activity recognition experiments using Opportunity dataset. From the experiments, we confirm the strong demands for rich sensor environments for activity recognition. On future works, we will continue to work on opportunity dataset using HASC Tool for the activity recognition. We will try to use more advanced features and more accurate learning data, the activity recognition rate might be improved.

Table1. Confusion Matrix of S2-ADL3

%	Stand	Walk	Sit	Lie
Stand	57.32	36.89	5.79	0.00
Walk	2.36	97.28	0.36	0.00
Sit	0.00	0.37	99.63	0.00
Lie	0.00	0.00	0.00	100.00
Overall	88.56			

Table2. Confusion Matrix of S3-ADL3

%	Stand	Walk	Sit	Lie
Stand	90.37	8.69	0.93	0.00
Walk	31.72	68.28	0.00	0.00
Sit	2.61	0.00	94.85	2.54
Lie	0.00	0.00	1.41	98.59
Overall	88.02			

REFERENCES

1. Kawaguchi, N., Ogawa, N., Iwasaki, Y., Kaji, K., Distributed Human Activity Data Processing using HASC Tool, in Proceedings of 13th ACM International Conference on Ubiquitous Computing, pp.603-604, 2011.
2. HASC Tool Project Website: http://sourceforge.jp/projects/hasc/
3. Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten, The WEKA Data Mining Software: An Update. *SIGKDD Explorations, Volume 11, Issue 1.*(2009).
4. Keng-hao Chang, Mike Y. Chen, and John Canny, "Tracking Free-Weight Exercises", UbiComp 2007. Ubiquitous Computing, pp. 19-37 (2007).
5. *Seon-Woo Lee and Kenji Mase. Activity and location recognition using wearable sensors. IEEE Pervasive Computing, 1(3):24–32, (2002)*
6. Ling Bao and Stephen S. Intille, "Activity Recognition from User-Annotated Acceleration Data", Pervasive 2004 , pp. 1-17 (2004)