A Wrist-Type Fall Detector with Statistical Classifier for the Elderly Care

Chankyu Park[1], Jaehong Kim[1], Joo-chan Sohn[1] and Ho-Jin Choi[2]
[1] Dept. of Robot/Cognitive System Research,
Electronics & Telecommunication Research Institute (ETRI), Daejeon, Korea
[e-mail: parkck@etri.re.kr]
[2] Dept. of Computer Science,
Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea
[e-mail: hojinc@kaist.ac.kr]
*Corresponding author: Ho-Jin Choi

Received April 4, 2011; revised July 15, 2011; revised August 25, 2011; accepted September 20, 2011; published October 31, 2011

Abstract

Falls are one of the most concerned accidents for elderly people and often result in serious physical and psychological consequences. Many researchers have studied fall detection techniques in various domain, however none released to a commercial product satisfying user requirements. We present a systematic modeling and evaluating procedure for best classification performance and then do experiments for comparing the performance of six procedures to get a statistical classifier based wrist-type fall detector to prevent dangerous consequences from falls. Even though the wrist may be the most difficult measurement location on the body to discern a fall event, the proposed feature deduction process and fall classification procedures shows positive results by using data sets of fall and general activity as two classes.

Keywords: Elderly care, wearable device, fall detection, accelerometers, statistical classifier, feature selection, feature reduction

The part of this paper was presented in the ICONI (International Conference on Internet) 2010, December 16-20, 2010, Philippines. This work was supported by the R&D program of the Korea Ministry of Knowledge and Economy (MKE) and the Korea Evaluation Institute of Industrial Technology (KEIT) [008-KI001836, Development of mediated Interface Technology for HRI]. This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0028631).

DOI: 10.3837/tiis.2011.10.004

1. Introduction

Falls are prevalent and dangerous accidents among the elderly people. Over a third of them over age 65 years fall each year [1]. For elderly over 85years, it increases to more than half [2]. References [3][4] assessed that 10% of falls make it the main cause of serious impairments to the health and lifestyle of the elderly group. In many cases, the early detection of fall is very important to rescue the subjects and avoid the severe consequences and significant medical costs.

Recently, many researches to detect fall effectively and to reduce the cost caring the subject have been reported in the widespread domain. With the enhancement of wireless networks and low-power MCU(Microcontroller Unit) technology, wearable fall detectors using inertial based sensors have been released and are capable of sending an alarm to caregivers or medical center automatically. These devices are small, inexpensive, easy to use, and can be worn at various body locations such as waist, chest, or back and have more merit than the vision based devices, in terms of privacy and blind spot problems [5][6][11].The elderly people may well accept a wrist-type device as a type of fall detector in many situations, since most people have no aversion about wearing a wrist.

However, the performance of fall detection algorithms operating at wrist-type devices is generally less than that applied by other body positions, because the arm and hand has six degrees of freedom, and always moves randomly during a daily life. Eventually, wrist-type fall detectors will have some possibility to give rise to many false alarms [7]. Most of the existing wrist-type fall detectors used threshold-based fall detection algorithms, which rely on heuristic methods, and are relatively simple [7][8]. Moreover, the number of subjects and their data set number of falls and general activities conducted by them are generally less. To make the fall detector to be robust and generalized to various activities, it should be trained by data sets conducted for many subjects and need to be implemented by using statistical or machine learning based classifier.

However, most of classification techniques generally require lots of computation, because they require many data sets having high dimension of a feature vector to represent characteristics of fall and general activity. In addition to ensure real-time operation of the fall detection algorithm in wrist-type low-power devices, it need to be simple and less computation, since a typical low-power MCU has low clock speed while it consumes less battery. Therefore, one of the candidates to reduce computation is to reduce dimension of a feature vector in a statistical based classifier, but it also must not degrade classification performance to some extent.

In this paper, we investigate the feasibility of using various statistical classifiers to detect fall events from daily activities and propose the efficient method to reduce a lot of computation in order to ensure real-time execution in wrist-type device having low-power MCU. The rest of the paper is organized as follows. In Section 2, we review our background work related to our fall detection problem and describe our systematic feature analysis and classification modeling approaches in detail. In Section 3, we present our experimental results using well defined six procedures for evaluating both feature analysis techniques and classifiers performance. Finally, we conclude our paper and discuss some possible future works in Section 4.

2. Background and Our Proposed Approach

2.1 Overview

We investigate the feasibility of using various statistical classifiers to detect fall events from daily activities and propose the efficient method to reduce a lot of computation in order to ensure real-time execution in wrist-type device having low-power MCU. **Fig. 1** introduces a systematic approach proposed in this study, which is composed of three major stages; (1) measurement for preprocessing 3-axis accelerometer signal and feature definition for extracting major features to represent the characteristics of accelerometer signal effectively; (2) feature analysis for reducing feature dimension and classifier modeling for preventing overfitting and complexity;(3) evaluation of performance measure for comparative criterion. From the following sections, we present detail description and processes of each stage

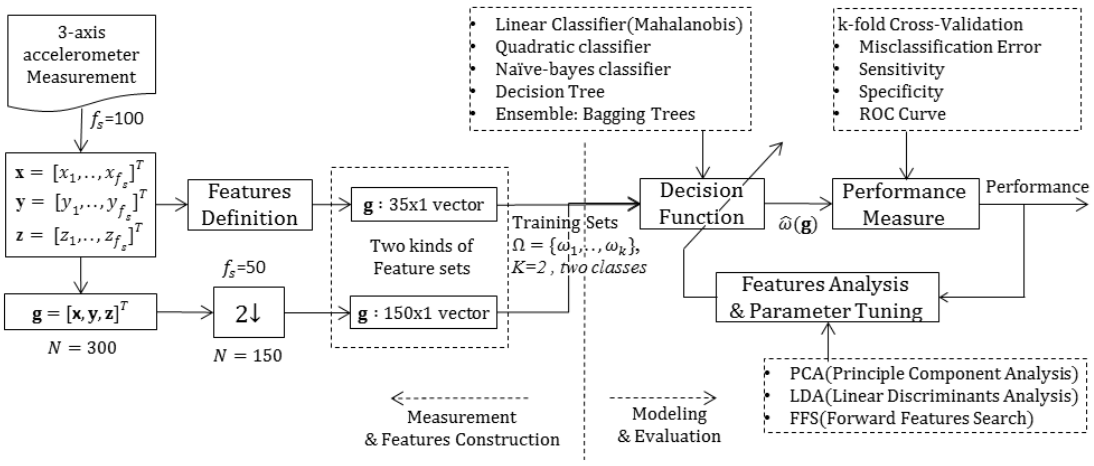

3-axis accelerometer Measurement
f_s=100
$\mathbf{x} = [x_1,..,x_{f_s}]^T$
$\mathbf{y} = [y_1,..,y_{f_s}]^T$
$\mathbf{z} = [z_1,..,z_{f_s}]^T$
Features Definition
$\mathbf{g}$: 35x1 vector
Two kinds of Feature sets
$\mathbf{g}$: 150x1 vector
$\mathbf{g} = [\mathbf{x}, \mathbf{y}, \mathbf{z}]^T$
$N = 300$
f_s=50
2↓
$N = 150$
Training Sets
$\Omega = \{\omega_1,..,\omega_k\}$,
K=2 , two classes
Linear Classifier(Mahalanobis)
Quadratic classifier
Naïve-bayes classifier
Decision Tree
Ensemble: Bagging Trees
Decision Function
$\hat{\omega}(\mathbf{g})$
k-fold Cross-Validation
Misclassification Error
Sensitivity
Specificity
ROC Curve
Performance Measure
Performance
Features Analysis & Parameter Tuning
PCA(Principle Component Analysis)
LDA(Linear Discriminants Analysis)
FFS(Forward Features Search)
Measurement & Features Construction
Modeling & Evaluation

Fig. 1. Overall our approach to design, modeling and evaluation for making a fall detector

2.2 Measurement and Features Definition

We conducted experiments to gather data sets of fall and general activity elicited by subjects wearing the wrist-type device for defining features and learning five types of classifier. **Fig. 2** shows wrist-type fall detector that includes a 3-axis accelerometer and an RF chip to transmit acceleration data to host computer for collecting them. In this study, 19 activity patterns were chosen to make a classifier to be more robust against false alarms, and simulated trials which are trying to play similar fall actions and other general activities were performed by 21 volunteers(ages range: 20 ages-5 people, 30 ages-9 people, 40 ages-6 people, 50 ages-1 people). The subjects consist of 16 male and 5 female. The selected lists of various activity patterns we adopted are categorized by characteristics of activity as shown in **Table 1**.

Under supervision of a researcher, subjects were asked to repeat three time trials for each pattern over 19 activity patterns, while only each hand pattern was repeated ten times. We grouped 19 patterns to two classes such as fall (4 patterns) vs. general activity (15 patterns) and then labeled each record into training data sets as two classes for typical binary classification. Therefore, we acquired 270 learning sets of fall and 2,800 learning sets of general activity.

In this study, we were only interested in features derived from the MEMS 3-axis accelerometer with ±6g range, which is embedded in the wrist-type fall detector. Many researchers reported that falls mostly happened within 0.8 seconds [6][9]. To accept this fall property, we used that the sliding window corresponds to one second for catching a fall event at sampling rate of 100 Hz. The sliding window shifted over the data without overlapping.

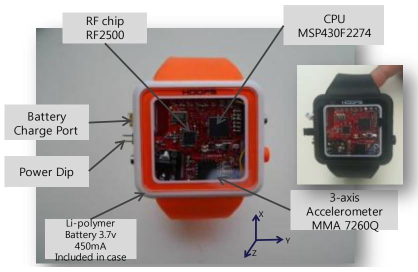
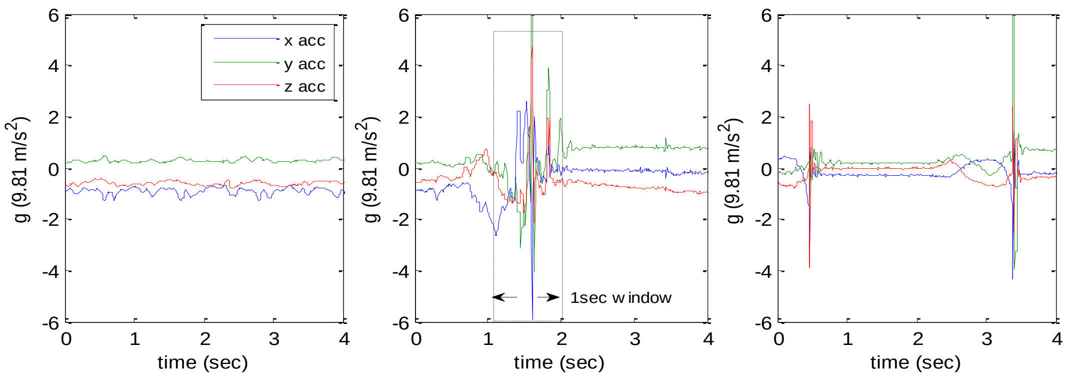


Fig. 2. Wrist-Type Fall Detector

Table 1. Categorized activity patterns

Fall patterns	Transient Patterns	Dynamic Patterns	Static Patterns
Fall-Forward	Sit-to-Stand	Walking	Standing
Fall-Back	Stand-to-Sit	Running	Sitting
Fall-Left	Sit-to-Lying	Upstairs	Lying
Fall-Right	Lying-to-Sit	Downstairs	
Hand Patterns			
Clapping	Stroking	Wrench	Hit shock



(a) (b) (c)

Fig. 3. Three color lines of three graphs (a)~(c) mean 3-axis accelerometer signals from the watch device which is put on subjects. (a) shows a pattern of a walking person, (b) shows a pattern of a falling person and (c) shows a patterns of a person who is clapping

Fig. 3 presents three kinds of 3-axis accelerometer signal patterns in the 19 patterns as an instance. In the case of walking pattern in Fig. 3-(a) , waveforms of signals are not changed abruptly and amplitude of signals is geneally restricted to the level within $\pm 2g$. Fig. 3-(b) shows one of the typical fall patterns in the training set and we can extract a fall event within a window at most cases. To differentiate fall patterns from similar shocks occurred by hand, we collected various patterns which are related with hand. These hand patterns are used to train classifiers to be more reliable for reducing error about false positive.

In order to investigate the effect of different size of dimensions and characteristic of features, we include all possible candidates of features and then we can try feature analysis process like feature selection and transformation techniques. We define two kinds of feature vectors. For the first type of feature vector **g_35** [35x1] described in **Fig. 1**, we computed the magnitude of the mean, variance, energy, maximum, minimum, the pairwise correlation of the the acceleration in *x*, *y* and *z* direction as well as the magnitude of the norm mean, norm variance, norm maximum, norm minimum, so that a **g_35** feature vector was created. We define clearly each feature's defintion of a **g_35** feature vector as shown in **Table 2**.

- feature vector **g**, measurement vectors for 3-axis accelerometer raw signal **x**,**y**,**z;** f_s is sampling frequency (100Hz), the dimension N of a feature vector **g**, is 35 such that $\mathbf{g} = [g_1,...,g_N]^T, \mathbf{x} = [x_1,...,x_{f_s}]^T, \mathbf{y} = [y_1,...,y_{f_s}]^T, \mathbf{z} = [z_1,...,z_{f_s}]^T, \Omega = \{\omega_1, \omega_2\}$ for K=2 , ω_1 ='fall', ω_2 ='general activity'.

Table 2. Each element description of **g** vector

Feature Vector $\mathbf{g}\_35 = [g_1,...,g_N]^T \; N = 35$
$[g_1, g_2, g_3]^T = [\hat{\mu}_\mathbf{x}, \hat{\mu}_\mathbf{y}, \hat{\mu}_\mathbf{z}]^T$ with $\hat{\mu}$ is sample mean
$[g_4, g_5, g_6]^T = [\hat{\sigma}_\mathbf{x}, \hat{\sigma}_\mathbf{y}, \hat{\sigma}_\mathbf{z}]^T$ with $\hat{\sigma}$ is sample variance
$[g_7, g_8]^T = [\hat{\mu}_\mathbf{m}, \hat{\sigma}_\mathbf{m}]^T$ for $\mathbf{m} = [m_1,...,m_{f_s}]^T$, $m_k = \sqrt{x_k^2 + y_k^2 + z_k^2}$, $1 \le k \le f_s$
$[g_9, g_{10}, g_{11}]^T = [\max_\mathbf{x}, \max_\mathbf{y}, \max_\mathbf{z}]^T$ for $\max_\mathbf{v} = \max_k \{\mathbf{v}\}, \; 1 \le k \le f_s$
$[g_{12},..., g_{15}]^T = [\min_\mathbf{m}, \min_\mathbf{x}, \min_\mathbf{y}, \min_\mathbf{z}]^T$ for $\min_\mathbf{v} = \min_k \{\mathbf{v}\}, \; 1 \le k \le f_s$
$[g_{16},..., g_{27}]^T = [\xi_{1,2}, \xi_{1,3}, \xi_{2,3}, \xi_{4,5}, \xi_{4,6}, \xi_{5,6}, \xi_{9,10}, \xi_{9,11}, \xi_{10,11}, \xi_{13,14}, \xi_{13,15}, \xi_{14,15}]^T$ with $\xi_{i,j} = \frac{g_i + g_j}{g_i^2 + g_j^2}$
$[g_{28}, g_{29}, g_{30}]^T = [\text{median}_\mathbf{x}, \text{median}_\mathbf{y}, \text{median}_\mathbf{z}]^T$ for $\text{median}_\mathbf{v} = \text{median}(\mathbf{v})$
$[g_{31}, g_{32}]^T = [\sqrt{\sum \text{diag}(\text{cov}(\mathbf{x},\mathbf{y}))}, l_m]^T$ with $l_m = n(\{m_k \mid m_k > 3.0\}) / f_s, 1 \le k \le f_s$
$[g_{33}, g_{34}, g_{35}]^T = [\hat{\mu}_\mathbf{m}^{t-1}, \hat{\sigma}_\mathbf{m}^{t-1}, \max_\mathbf{m}^{t-1}]^T$ with t -1 = index of the pervious window frame

For the second type of feature vector **g_150** [150x1] described in Fig. 1, we just included raw data of 3-axis accelerometer sensor for 1 seconds and a [300x1] raw feature vector is created. On the stage of feature reduction or classification, we need to compute covariance matrix and inverse matrix of training sets and the input vector must be positive definite matrix. Also, the size of each class training set should be larger than the size of feature dimension. In our cases, fall training set is 270 records and the size of raw feature vector is 300, so that we have to do downsampling raw data to 2 factor to be 150 dimension vector.

2.3 Feature Reduction Analysis

Reducing the dimensionality of features is important in statistical learning. In some cases, the dimension *N* of a feature vector **g** can be very high. Many elements of **g** can be redundant or even irrelevant in terms of the classification. We have to be careful to decide the dimension of a feature vector according to two major reasons: The first is that the computational complexity comes to be too large. A linear classifier requires in the order of *KN* operations. A quadratic classifier needs about KN^2 operations, for example *K*=1, *N*=200, 200^2(a 200x200 image). The second is that increasing the dimension ultimately results in decreasing classification performance.

To address both problems, one can apply two main approaches to reduce features: feature selection and feature transformation. Feature selection algorithms select a subset of features from the original feature set; feature transformation methods transform data from the original high-dimensional feature space to a new space with reduced dimension. Feature transformation methods use dimension reduction algorithms to be achieved by computing an optimal subset of predictive features measured in the original data. Followings introduce three kinds of feature selection and transformation methods used in this study.

Principle Component Analysis (PCA)

PCA can be used as one of the feature transformation methods, so that set of measurements of possibly correlated features into a set of features of uncorrelated features called principal components. PCA can find optimal linear transformation which can best represent the data in a minimum mean square error sense. PCA is also called as a kind of unsupervised clustering, since it does not take into account the class that features belong to. We have $\mathbf{y} = \mathbf{W}_D^T\mathbf{g}$ where the NxD matrix $\mathbf{W}_D^T\mathbf{g}$ transforms the feature space $\Re^N$ onto a reduced space $\Re^D$ ($N >> D$) and $\mathbf{y}$ is a good estimate of $\mathbf{g}$ even with the lower dimension of $\mathbf{y}$ [10]. $\mathbf{W}_D$ can be written as:

$$\mathbf{W}_D = \arg\min_{\mathrm{W}}\{\mathrm{E}[\|\hat{\mathbf{g}} - \mathbf{g}\|^2]\} \tag{1}$$

where one can find $\mathbf{W}_D$ to minimize reconstruction error such that $\mathbf{W}_D$ estimate $\hat{\mathbf{g}}$ as least square error sense. Under the condition that the elements of $\mathbf{y}$ must be uncorrelated, we can use diagonal matrix characteristics to solve the (1). This can be described as:

$$\mathbf{W}_{D_{opt}} = \arg\max_{\mathrm{W}}\{\mathrm{trace}(\mathbf{W}^T\mathbf{C}\mathbf{W})\},\ \text{where } \mathbf{C} = \mathbf{g}\mathbf{g}^T \tag{2}$$

$$\mathrm{trace}(\mathbf{W}^T\mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T\mathbf{W}^T),\ \text{where } \mathbf{C} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^T, \boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots \lambda_D) \text{ by SVD} \tag{3}$$

We use SVD to make $\mathbf{W}^T$ to be a diagonal matrix so that optimally the transformation matrix can be composed of D eigenvectors as:

$$\mathbf{W}_{D_{opt}} = \mathbf{V} = (\mathbf{e}_1 | \mathbf{e}_2 | \ldots | \mathbf{e}_{\hat{D}}),\ \text{where } \mathbf{e}_1 \text{ is } i\text{th eigen vector} \tag{4}$$

$\mathbf{W}_{D_{opt}}$ transforms $\mathbf{g}$ into $\hat{D}$ dimensional new coordinate which makes the largest variance of the data to lie on the first $\mathbf{e}_1$ axis, the second largest variance on the second $\mathbf{e}_2$, lastly $\mathbf{e}_{\hat{D}}$ [10]. PCA retains principal variability in the data and do not take into account small variability.

Linear Discriminant Analysis (LDA)

LDA [3] also seeks for the best features in the reduced space, but it considers each class where features belong to and so that it can get the features that best discriminate among classes. The basic idea of a criterion function to evaluate those vectors to be best discriminable features is to maximize distance of between-class while minimizing distance of within-class. LDA creates a linear combination of these which yields the largest mean differences between the desired classes [10]. We define two measures: one is within-class scatter matrix, as given by

$$\mathbf{S}_W = \frac{1}{N_S}\sum_{k=1}^{K}\sum_{n=1}^{N_k}(\mathbf{g}_{k,n}-\hat{\boldsymbol{\mu}}_k)(\mathbf{g}_{k,n}-\hat{\boldsymbol{\mu}}_k)^T \text{ where } \hat{\boldsymbol{\mu}}_k = \frac{1}{N_k}\sum_{n=1}^{N_k}\mathbf{g}_{k,n} \quad (5)$$

where $\mathbf{g}_{k,n}$ is the *n*th sample of class *k*, $\hat{\boldsymbol{\mu}}_k$ is the mean of class *k*, *K* is the number of classes, N_k is the number of samples in class *k*, and N_s is the total number of samples. The other is between-class scatter matrix:

$$\mathbf{S}_b = \frac{1}{N_S}\sum_{k=1}^{K}(\hat{\boldsymbol{\mu}}_k-\hat{\boldsymbol{\mu}})(\hat{\boldsymbol{\mu}}_k-\hat{\boldsymbol{\mu}})^T \text{ where } \hat{\boldsymbol{\mu}} = \frac{1}{N_s}\sum_{n=1}^{N_S}\mathbf{g}_n \quad (6)$$

where $\hat{\boldsymbol{\mu}}_k$ is the mean of all classes

$\mathbf{W}_{opt}$ that we want to get can be written as:

$$\mathbf{W}_{opt} = \arg\max_{\mathrm{W}} J(\mathbf{W}) \quad (7)$$

Using (5), (6) and maximizing $\mathbf{S}_b$ while minimizing $\mathbf{S}_W$, we can induce *J*(**W**)can be represented to:

$$J(\mathbf{W}) = \frac{\mathbf{W}^T\mathbf{S}_b\mathbf{W}}{\mathbf{W}^T\mathbf{S}_W\mathbf{W}} \quad (8)$$

In case that the number of classes is *K*=2, we can get simplified **W** from (7) and (8)

$$\mathbf{W} = \arg\max_{\mathrm{W}}\left\{\frac{\mathbf{W}^T\mathbf{S}_b\mathbf{W}}{\mathbf{W}^T\mathbf{S}_W\mathbf{W}}\right\} = \mathbf{S}_W^{-1}(\hat{\boldsymbol{\mu}}_1-\hat{\boldsymbol{\mu}}_2) \text{ where } K=2, \text{two classes} \quad (9)$$

A characteristic of LDA is that the **y** dimension of the feature space found through $\mathbf{y}=\mathbf{W}^T\mathbf{g}$ will not exceed *K*-1, where *K* is the number of classes. This follows from (6), which shows that $\mathbf{S}_b$ is the sum of *K* outer products of vectors. Therefore, $rank(\mathbf{S}_b)$ cannot exceed *K*-1. As a result, the number of nonzero eigenvalues of $\mathbf{S}_b$ cannot exceed *K*-1the dimension need at least *N*+*K* samples to guarantee that $\mathbf{S}_W$ does not become singular where *N* is dimension of original feature space.

FFS : Forward Feature Selection

FFS is one of the popular algorithms in sequential feature selection. When one considers meaning of features which is significant and the goal of feature reduction which is to identify a dominant sub-feature set, feature selection techniques are more desirable than feature transformation. Selection criteria usually involve the minimization of a specific performance measure for models fit to different subsets. The algorithms search for a subset of predictors that optimally model measured responses, subject to constraints such as added or removed features and the size of the subset. In the most cases, one use misclassification rate as a criterion.

Let $\mathrm{F}(N)=\{\mathrm{g}_n \mid n=0,...,N\text{-}1\}$ be the set with elements from original feature vector g and $\mathrm{F}_j(D)=\{\mathrm{y}_d \mid d=0,...,D\text{-}1\}$ be a subset with of *D* elements taken from **g** [10]. Let $J(\mathrm{F}_j(D))$

be a performance measure related to the subset $\mathrm{F}_j(D)$. In order to find a subset $\hat{\mathrm{F}}(D)$, this subset surpasses all other subsets with dimension *D*:

$$\hat{\mathrm{F}}(D) = \mathrm{F}_i(D) \text{ with} : J(\mathrm{F}_i(D)) \geq J(\mathrm{F}_j(D)) \text{ for all } j \in \{1,\dots,\mathrm{q}(D)\}, \mathrm{q}(D) = \begin{pmatrix} N \\ D \end{pmatrix} \tag{10}$$

In FFS, features are sequentially added to an empty candidate set until the addition of further features does not decrease the performance measure, and sequential searches proceed in only one direction, always growing the candidate set, because a complete comparison of the criterion value over all $\mathrm{q}(D)$ subsets is impractical in general [10]. Therefore, although efficient search algorithms are existed, they are suboptimal.

2.4 Modeling Classification Models

In this section we investigate various classification models and techniques which methods is effective or not on our training set. The model generated by a learning algorithm should both fit the input feature vector well and correctly estimate the class labels of records it has never seen before. Therefore, the main goal of the learning algorithm is to build models with good generalization capability.

In addition, we have to ensure that classification models are feasible to be implemented into embedded H/W having little computing power. From preliminary research, to avoid very complex and heavy algorithms, we suggest very compact and simple classification techniques to be used in this study such as Bayes classification based on general distributed probability density with uniform cost function, decision trees based classification and finally a bagging technique to combine weak classifier.

Quadratic Classifier

In Bayes Classification, when we suppose that Bayes classification have uniform cost function and the conditional probability densities are modeled as normal Gaussian distribution, it can be minimum error rate classifier. For a further development of Bayes classification, the feature vectors coming from class ω_k are normally distributed with expectation vector $\boldsymbol{\mu}_k$ and covariance matrix $\mathbf{C}_k$. Lets $\hat{\omega}(.) : \Re^N \to \Omega$ be a Bayes classifier [10]. We can induce $\hat{\omega}(.)$ from MAP(maximum a posterior) and definition of conditional normal probability density, and can be shown as:

$$\hat{\omega}(\mathbf{g}) = \omega_i, \text{ where } i = \arg\max_{\kappa=1,..K} \{\mathrm{w}_k + \mathbf{g}^T \mathbf{w}_k + \mathbf{g}^T \mathbf{W}_k \mathbf{g}\} \tag{11}$$

where $\mathrm{w}_k = -\ln|\mathbf{C}_k| + 2\ln P(\omega_k) - \hat{\boldsymbol{\mu}}_k^T \mathbf{C}_k^{-1} \hat{\boldsymbol{\mu}}_k, \mathbf{w}_k = 2\mathbf{C}_k^{-1}\boldsymbol{\mu}_k, \mathbf{W}_k = -\mathbf{C}_k^{-1}$. So, it is called quadratic classifier because $\mathbf{W}_k$ is existed as a quadratic form.

Linear Classifier

We introduce another case in which the covariance matrices do not depend on the classes, i.e. $\mathbf{C}_k = \mathbf{C}$ for all $\omega_k \in \Omega$ .Therefore, the class information is changed to depend on the expectation vectors $\boldsymbol{\mu}_k$ only.

$$\hat{\omega}(\mathbf{g}) = \omega_i, \text{ where } i = \arg\max_{\kappa=1,..K} \{\mathrm{w}_k + \mathbf{g}^T \mathbf{w}_k\} \tag{12}$$

where $\mathrm{w}_k = \ln P(\omega_k) - \hat{\boldsymbol{\mu}}_k^T \mathbf{C}_k^{-1} \hat{\boldsymbol{\mu}}_k, \mathbf{w}_k = 2\mathbf{C}^{-1}\boldsymbol{\mu}_k$. A decision function which has the form of (12) is linear [10]. The corresponding classifier is called a linear classifier or Mahalanobis distance based classifier.

Naïve Bayes Classifier

We assume that each feature $\mathbf{g}_i$ is conditionally independent of every other feature $\mathbf{g}_j$ for $i \neq j$. This means that $p(\mathbf{g}_i \mid \omega, \mathbf{g}_i) = p(\mathbf{g}_i \mid \omega)$ this is known as MAP decision rule. The assumption is not accurate and naive, so that it is called as Naïve Bayes classifier. The corresponding classifier is the function $\hat{\omega}_{MAP_{naive}}(\mathbf{g})$ defined as follows [10]:

$$\hat{\omega}_{MAP_{naive}}(\mathbf{g}) = \arg\max_{\omega \in \Omega} \{ p(\mathbf{g} \mid \omega) P(\omega) \} \tag{13}$$

$$= \arg\max_{\omega \in \Omega} \left\{ P(\omega) \prod_{i=1}^{N} p(\mathbf{g}_i \mid \omega) \right\} \tag{14}$$

Despite of the strong independence assumptions, the naive Bayes classifier has several properties: each feature distribution can be independently estimated as a one dimensional distribution. This helps to relieve from computation complexity in the high dimensional feature space.

Decision tree

Decision tree classifier is a nonparametric and nonlinear approach for constructing classification tree models. In other words, it does not require any prior assumptions comparing the type of probability distributions satisfied by the class and other features. Constructing decision trees are computationally inexpensive, making it possible to quickly construct models even when the training set size is very large. Additionally, after a decision tree has been built, classifying a test sample of feature vector is extremely fast, with a worst-case complexity of $\mathrm{O}(t)$, where t is the maximum depth of the tree [10].

However, if the feature set contains many irrelevant features that are not useful for the classifier, then some of the irrelevant features may be included during the tree-growing process, which results in a decision tree that is larger. Feature selection techniques can help to improve the accuracy of decision trees by eliminating the irrelevant features during feature analysis. Pruning is also strongly recommended to maximize over-fitting while minimizing the dimension of original feature space.

Bagging Trees

Bagging Trees is a kind of ensemble learning which is to bag a weak learner as adopting a decision tree on a feature set and generate many bootstrap samples of the training feature set and grow decision trees on these bootstraps. We can get each bootstrap samples by randomly selecting N_S observations out of N_S with replacement, where N_S is the number of feature set. Training is done by a classifier using each bootstrap sample.

For classification, bagging trees returns majority vote on the classification results which are averaged over responses from individual trees [10]. Bagging has an advantage to improve performance for unstable classifiers which vary significantly with small changes in the feature set. Likewise basic decision tree, feature selection techniques are also developed to minimize the redundant features which are not useful while growing bootstrap to larger size. We use this method to select useful feature set while satisfying feasible generalization error.

3. Experiments

3.1 Performance Measures and Experiments Procedure

This section introduces some of the methods commonly used to evaluate the performance of a classifier and then various combination procedures of feature analysis and classification model conducted in this experiment.

First, a good classification model must not only fit the training data well, it must also accurately classify records it has never seen before. This is important since a classification model that fits the training data too well can have a worse generalization error than a model with a higher training error, since the model must be overfitting to the training data. In this paper, we used 10-fold cross-validation method for estimating the generalization error of a chosen model in each procedure. The estimated cross-validation error helps learning algorithm to find a model of the precise complexity that is not sensitive to overfitting.

We can particularly compute four performance measures such as misclassification error, sensitivity, specificity and ROC (the receiver operating characteristic) curve because the training data has two classes and thus it results in binary classification problem.

Table 3 shows their relationships among terms. Sensitivity is to measure the proportion of actual falls that are correctly identified as such, and specificity is to measure the proportion of actual general activity that are correctly identified as such. We maintain that one of the most important measures is sensitivity, because people who are falling can result in being dangerous situations when false negative is high. Consequently, the possibility to raise an alarm may be low. Sensitivity is a good measure to show how much the system is sensitive to these situations.

Table 3. Performance measure for binary classification: confusion matrix

		True Labels	
		Positive (Fall)	**Negative** (General Activity)
Estimated Labels	**Positive** (Fall)	True Positive (TP)	False Positive (FP)
	Negative (General Activity)	False Negative (FN)	True Negative (TN)
Accuracy = TP + TN Misclassification Error = 1- Accuracy		Sensitivity = TP/(TP+FN)	Specificity = TN/(FP+TN)

Second, we create six combination procedure using three feature analysis techniques and five classification techniques as described previous section, apply these combinations to two kinds of training data, and then finally evaluate the result of them. The six evaluation procedures are presented in **Table 4** and we apply these methods sequentially.

When we followed the C1 procedure, we realized that the performance from C1 was poor and under our expectation when 150 training set is applied to both LDA and FFS except for PCA. We decided not to continue LDA and FFS analysis further, since FFS can provide good performance when features are composed of somewhat meaningful things, but a **g_150** vector is just raw data and do not have special meaning. In addition, FFS took too much time to evaluate [150x1] training set. However, PCA could be applied to the training data because it has a property to compact raw data to the principal components.

In the C3 procedure, we apply FFS method to **g_35** training set. To begin with FFS, evaluation criterion should be decided that which features are effective to the target classifiers. We use wrapper methods to use the performance of the chosen target classifier's learning algorithm to evaluate each candidate feature subset [9]. Therefore, we adopt linear, quadratic and naïve Bayes classifiers as target classifier's learning algorithm. In the C4 procedure, we apply LDA method to **g_35** training set as a method of feature reduction. LDA introduced in Section 2.3 has a property that it transforms feature vector to *K-1* reduced vector space. Consequently, we can get just one dimensional result vectors from [35x1]. Thus, we do not need iterative operations over the reduced feature dimensions like PCA. When we use linear, quadratic, naïve Bayes classifiers, MLE(Maximum likelihood estimation) is used as parameter learning technique.

Table 4. Combination of feature analysis and classification applied to this experiment

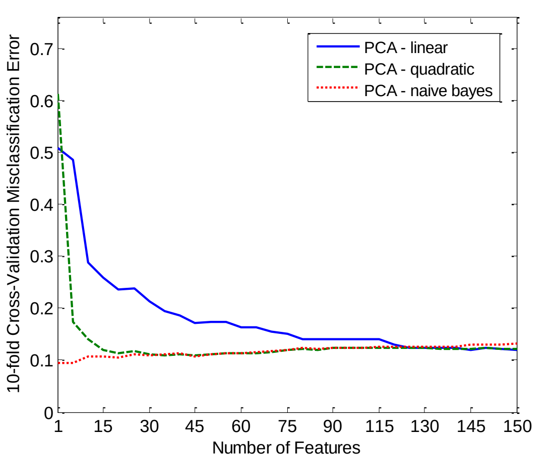
Cases	Description about stage sequences of each trial
C1	**g_150** feature vector → PCA → {linear, quadratic, navie_bayes}
C2	**g_35** feature vector → PCA → {linear, quadratic, navie_bayes}
C3	**g_35** feature vector → FFS(linear|quadratic|navie) → {linear, quadratic, navie_bayes}
C4	**g_35** feature vector → LDA → {linear-1, linear-2, quadratic, naive_bayes, DT}
C5	**g_35** feature vector → DT → pruning
C6	**g_35** feature vector → (BaggingTrees|feature_selection) → pruning

3.2 Experimental Results

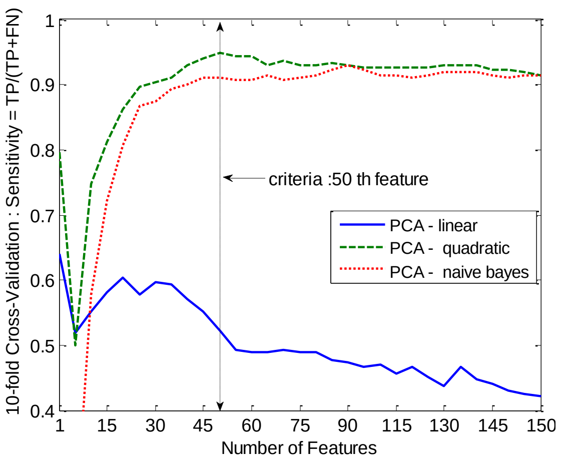
In this section we introduce our experimental results conducted according to six cases in **Table 4**.

C1 Procedure

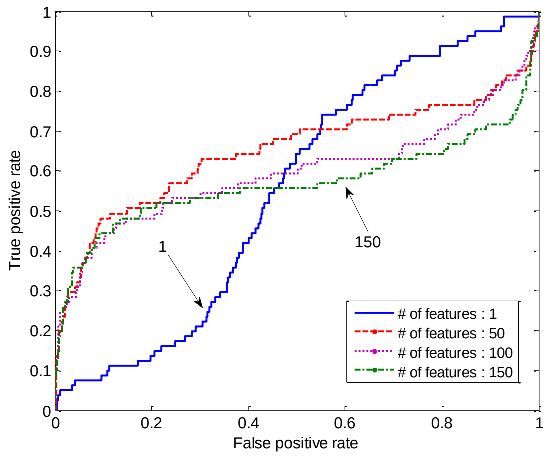
The results of the C1 procedure are shown in **Fig. 4**, where the x axis of **Fig. 4-(a)(b)** indicates the number of reduced features; the y axis represents respectively misclassification error and sensitivity and **Fig. 4-(c)(d)** shows respectively four ROC curves in the linear and naïve Bayes classifiers. In this procedure, linear classifier was set to Mahalanobis based type. When the number of features in **Fig. 4-(a)** increases from 1 to 35, misclassification error curves drop gradually representing that the error performance improved significantly with an increase in feature number. In terms of sensitivity, PCA-quadratic and PCA-naïve Bayes show feasible error performance except PCA-linear. However, overall misclassification errors are under 0.9 and also both procedures still require over 50 features although they reduce redundant features. In **Fig. 4-(a)**, in terms of ROC, C1.navie_bayes shows best performance when using 50 features. Consequently, PCA-quadratic and PCA-naïve Bayes cases are more reliable than the PCA-linear case according to the number of dimension of feature vector **g_150**, because sensitivity and specificity is high at the same time.



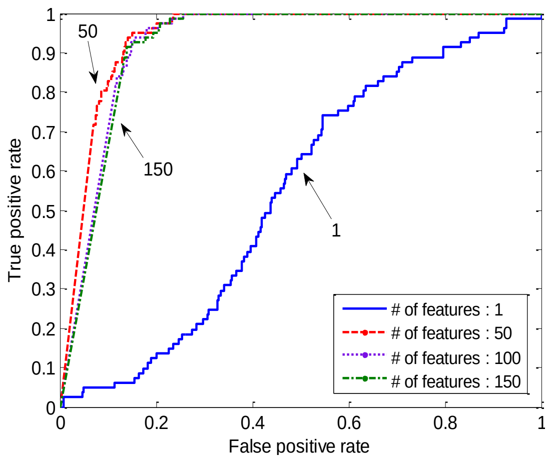
(a)



(b)



(c)



(d)

Fig. 4. C1 : **g_150** feature vector →PCA → {linear, quadratic, navie_bayes}

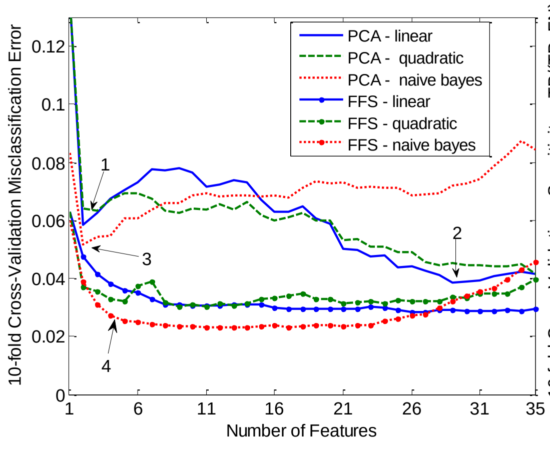
(a) Misclassification Error **(b)** Sensitivity **(c)** ROC curve: C1.linear **(d)** ROC curve : C1.naive_bayes

C2 and C3 Procedure

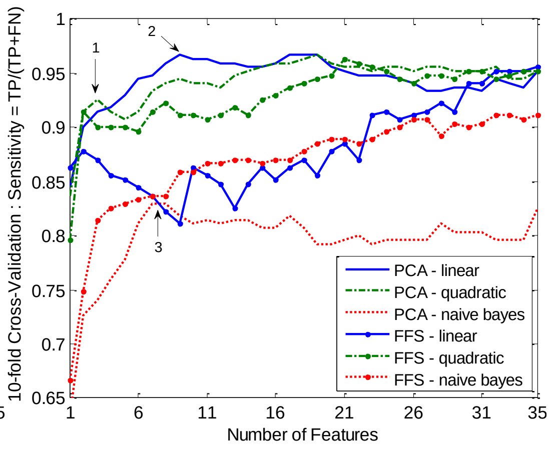
The results of the C2 and C3 procedure are shown in **Fig. 5**, where the x axis of all graphs indicates the number of reduced features; the y axis represents respectively misclassification error, sensitivity and specificity. We conducted PCA and FFS of feature reduction techniques to training set using 10-fold cross validation. Totally, there are different six cases to evaluate performance measures. In terms of misclassification error, **Fig. 5-(a)** shows that all FFS based classifications are better than the all PCA based classifications over the number of **g_35**, because the errors are usually under 0.04. In other words, accuracy is over 96%.

We present the four graph points to be acceptable as best point to minimize generalization error in misclassification error sense as shown in **Fig. 5-(a)**. Suggesting the continual increase of features over best points does not make a significant contribution to the increase of the sensitivity performance. Moreover, in terms of specificity, the results are similar to the results of previous misclassification error because those values over 0.97.

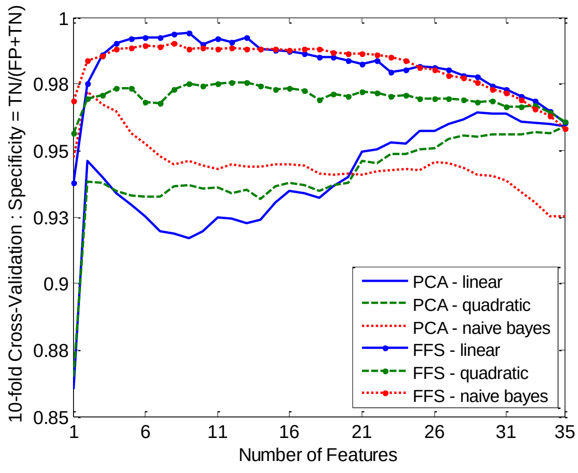
On the other hand, in terms of sensitivity, PCA-linear, quadratic cases outperform all FFS based classifications as shown in **Fig. 5-(b)**. FFS-linear shows that its sensitivity is fluctuate over x-axis. If one favors that sensitivity are more important than specificity while specificity is low relatively in fall detection, choosing point 1 or 2 in **Fig. 5-(b)** will be a good decision because those points shows good sensitivity as well as the low number of features such as 4 or 10. Additionally, if we choose point 2 of PCA-linear in **Fig. 5-(b)**, its computation complexity is very low and just *KD* order, where *D* is a reduced number of dimension and *K*=2.



(a)



(b)



(c)

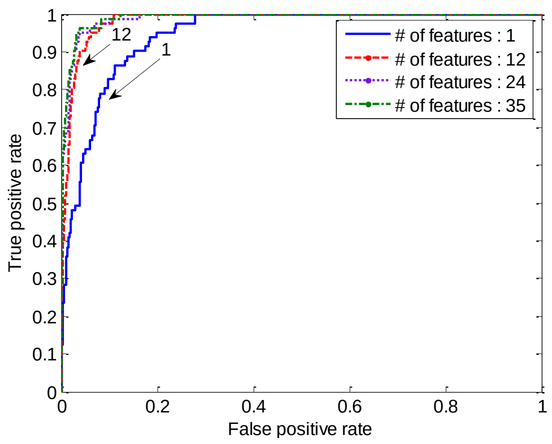
Fig. 5. C2: **g_35** feature vector → PCA → {linear, quadratic, navie_bayes},

C3: **g_35** feature vector → FFS(linear|quadratic|navie) → {linear, quadratic, navie_bayes}

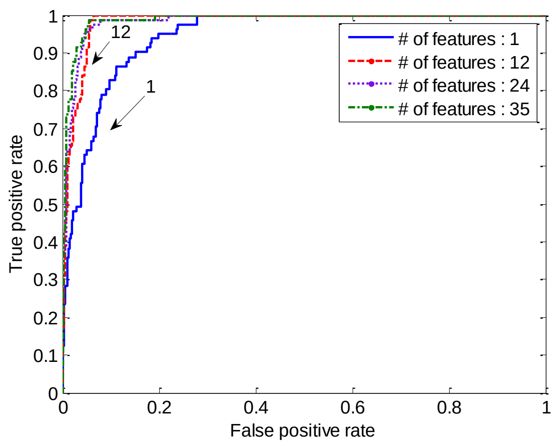
(a) Misclassification Error C2,C3 **(b)** Sensitivity C2,C3 **(c)** Specificity C2,C3

Fig. 6 shows four ROC curves to describe the results from C2-linear, quadratic, naïve Bayes and C3-linear respectively. As shown in the previous **Fig. 5-(b)**, PCA-linear case can be regarded as having reasonable performance if considering only sensitivity. However, we can know that FFS-linear has also good performance as much as PCA-linear as shown in **Fig.6-(a)(d)**. From these results, We can estimate that FFS based feature selection techniques C3 are useful and they can select meaningful features to the given training set.

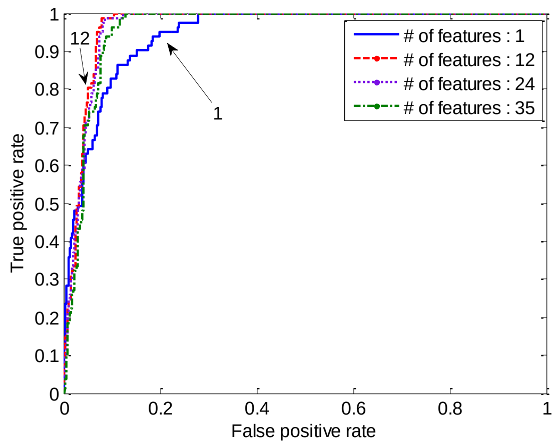
In case by FFS-linear case in **Fig. 6-(d)** shows best AUC(area under curve) when using 12 features. But, all cases of **Fig. 6-(a)(b)(c)** except for **(d)**, the ROC curves become flat to the y-axis when the false positive rate is decreased to the zero value and the true positive rate is abruptly decreased as well. This is just unavoidable situation so that trade-off can always be happening when one tries to do tuning classifiers.



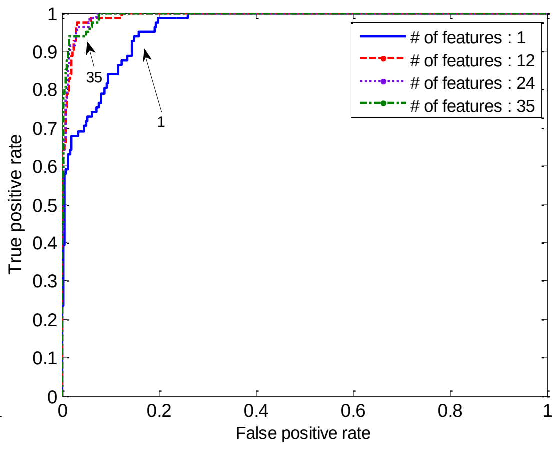
(a)



(b)



(c)



(d)

Fig. 6. C2: **g_35** feature vector → PCA → {linear, quadratic, navie_bayes},

C3: **g_35** feature vector → FFS(linear|quadratic|navie) → linear

(a) ROC C2-linear **(b)** ROC C2-quadratic **(c)** ROC C2-naive **(d)** ROC C3-linear

C4 Procedure

As described in Section 3.1, C4 procedure do not require exhausted iterative feature deduction process because LDA transform **g_35** feature vector to one dimensional result vectors. In this procedure, two types of linear classifier such as Mahalanobis and minimum distance based types were applied to LDA transformed training set. **Table 5** shows the accuracy, sensitivity and specificity results according to five types of classifiers by LDA feature transformation with [35x1]. Remarkably, all classifiers show good performance on average although the reduced feature dimension is just one dimension. The reason is that LDA is originally designed to get best discrimination between classes, while PCA is a kind of the data compacting methods as the best description of the data in this entirety.

In terms of sensitivity, linear classifier (Mahalanobis) achieves best performance while accuracy measure is lowest of them. Naïve-Bayes classifier achieves best performance in accuracy but sensitivity relatively is low. In terms of both computation complexity and performance measure, the classifier is one of the best candidates for deploying it to wrist-type embedded H/W. In addition, we can find that ROC curve of the linear classifier (Mahalanobis) shows good performance as shown in **Fig. 7**.

Table 5. **g_35** feature → LDA → {linear-1, linear-2, quadratic, naive_bayes, DT}

Classification	Accuracy	Sensitivity	Specificity
Linear-1 Minimum distance	0.972	0.889	0.980
Linear-2 Mahalanobis	0.948	0.956	0.945
Quadratic	0.956	0.922	0.960
naïve-Bayes	0.973	0.889	0.982
Decision Tree	0.961	0.800	0.978

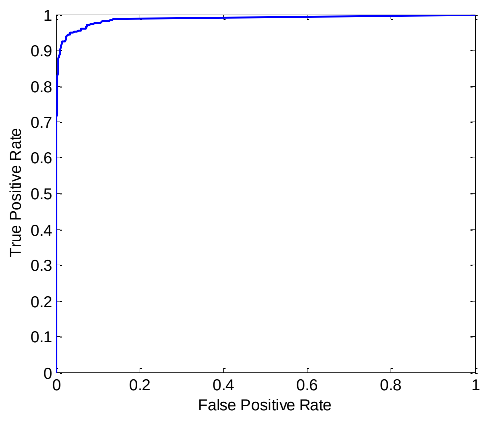
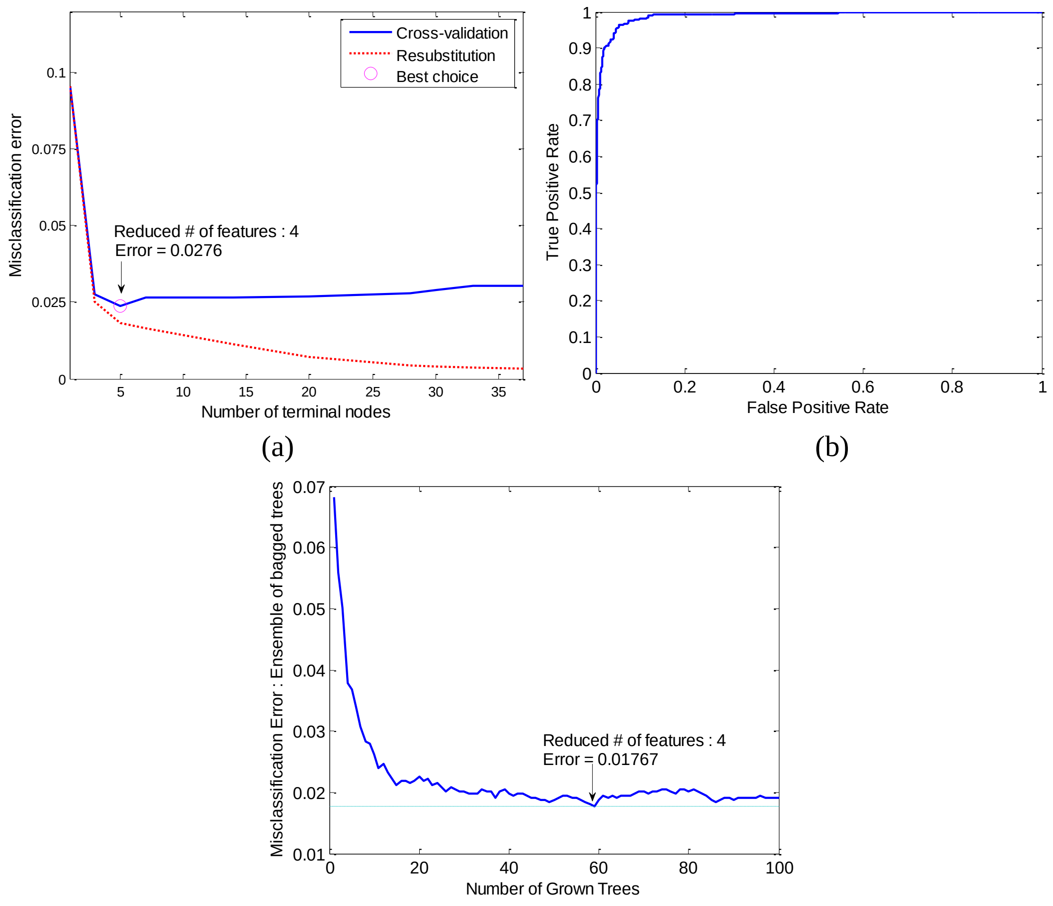


Fig. 7. C4: **g_35** feature → LDA → linear-2, ROC curve

C5 and C6 Procedure

Decision tree based classification is simpler than different other classification techniques while PCA and FFS feature reduction techniques requires exhaustive searching tasks. Decision trees has a powerful tool to be called pruning which is used to cut all nodes under tree level decided by cross-validation process. The results of decision tree classification are shown are shown in **Fig. 8-(a)(b)**, where the x axis of **Fig. 8-(a)** indicates the number of constructed tree node; the y axis indicates misclassification error. **Fig. 8-(a)** shows that re-substitution error is gradually decreased when the number of terminal nodes of decision trees is decreased. On the other hand, cross-validation error is decreased where the tree's node is near 5th terminal node and then again the error is increased over the 5th terminal node. This result gives us that training of the tree is getting over fitted to their training set. Therefore, the best candidate level is 5th terminal node level and pruning the level results in 0.0276 error (accuracy: 97.2%).

Furthermore we can get reduced four feature sets enable to construct new decision tree having the best performance of generalization error. We can know that these four features have discriminative power in this decision tree and feature selection of a decision tree is more effective than FFS. The selected four features are $\{g_4, g_{20}, g_{32}, g_{35}\}$and whole 31 features except for the four features are not necessary to training set. The preprocessing stage in embedded wrist-type H/W just computes the four features, and the next step is to apply simple binary rules of the decision tree to actual feature vector from measurement. In the **Fig. 8-(b)**, ROC curve of decision tree shows good performance as well.



(a) (b)

(c)

Fig. 8. C5: **g_35** feature vector → DT → pruning,

C6: **g_35** features vector → (Bagging Trees|feature_selection) → pruning

(a) Misclassification Error C5 (b) ROC C5 (c) Misclassification Error C6-naive

In order to compare decision tree's performance to other tree based techniques, we conducted C6 procedure using begging tree algorithm as shown in **Fig. 8-(c**), where the x-axis indicates the number of bagging trees(bootstraps). This result presents that misclassification error curves drop sharply when the number of grown trees increases from 1 to 20. But, overall error performance from 21 grown trees to 100 grown trees is maintained by the level of 20 grown trees. Bagging trees can be also tried to get reduced feature vectors equivalent to the four features produced by decision tree method. Consequently, the reduced features are the same of decision tree. We can get less misclassification error 0.01767 (accuracy: 98.2%) than decision trees, whereas Bagging trees require about 60 grown sub trees while decision tree requires one tree having 5 terminal nodes.

With C1~C6 procedure, we investigated the effect of various combination of feature anaysis and classification techniques. In the C1 procedure, it presented that **g_150** feature set is not suitable for fall detection, and at least feature set extracted from feature definition should be needed as described in section 2.2. In the C2 and C3 procedure, we can summary that FFS based feature selection is more reliable than PCA based on transformation through ROC curve analysis. In the C4 procedure, LDA-based on transformation shows generally good performance in the five kinds of classifiers. LDA-linear case of them is best, but this method must compute all 35 features and then **g_35** vector can be transformed to one dimension vector using LDA matrix **W**. Although feature dimension is reduced to one, it shoud require all 35 features in the preprocessing state. Finally, C5 and C6 also shows good results for the generalization error. Decision tree has more advantages than LDA-linear, since it just four four features and also do not require any transformation whereas LDA-linear requires.

To ensure an embedded online classifier algorithm to be real-time operation in a wrist-type device, we have to seek a minimal order of the feature vector satisfying maximal performance of recognition as much as possible, since the classifier should be simple and tightly depends on the dimension of feature vector as a matrix operation. Consequently, we realized that simple is best and both LDA-linear and decision tree method are the best candidate. The required real-time speed of the both methods in the embedded hardware should be 100Hz. We confirmed that real-time operation of both methods was performed in without delay in wrist-type fall detector.

4. Conclusion

In this paper, in order to reduce a lot of computation and to ensure real-time operation under a limited low-power MCU environment, we have demonstrated the feasibility by using feature reduction techniques on large data sets and high dimension of a feature vector. We observed that a reduced dimension of a feature vector could be applied to acquire the recognition rate over 90%. Moreover, we have checked that LDA-linear method and decision tree method show best performance results having just four features and simple decision rules. Besides improving the approach taken in this paper to a larger set of activities, we plan to classify multiple classes such as walking, running, lying, etc., as well as fall in future work. On the other issues, when we are labeling training data for each class, labeling itself is very exhausted task and requires a lot of cost, time and expertise of old people’s activities. Another topic is a problem not to collect actual old people's full and general activities. Although we have actual fall data from old people, most of them may be gathered using unsupervised approach. How can we add fall data which are gathered by unsupervised approach to the supervised training sets? We have a plan to research combining unsupervised data set and semi-supervised learning techniques.

References

[1] H. Luukinen, K. Koski, R. Honkanen, S. Kivelä, “Incidence of injury-causing falls among older adults by place of residence: a population-based study,” *Journal of the American Geriatrics Society* vol. 43, pp. 871-876, 1995. Article (CrossRef Link)

[2] A. J. Blake, K. Morgan, “Falls by elderly people at home:Prevalence and associated factors,” *Age and Ageing*, vol. 17, pp. 365-372, 1998. Article (CrossRef Link)

[3] G. Pérolle, D. Sánchez, M.I. Abarrategui, G. Eizmendi, C. Buiza, I. Etxeberria, J.J. Yanguas, "Fall Detection: Project of an Improved Solution," in *Proc. of 1st International Workshop on Tele-Care and Collaborative Virtual Communities in Elderly Care*, 2004. Article (CrossRef Link)
[4] K. Doughty, R. Lewis, A. McIntosh, "The Design of a Practical and Reliable Fall Detector for Community and Institutional Telecare," *Journal of Telemedicine and Telecare*, vol. 6, pp. 150-154, 2000. Article (CrossRef Link)
[5] M. Kangas, A. Konttila, I. Winblad, T. Jamsa, "Determination of simple thresholds for accelerometry-based parameters for fall detection," in *Proc. of 29th Annual International Conference on Engineering in Medicine and Biology*, pp. 1367-1370, 2007. Article (CrossRef Link)
[6] U. Lindemann, A. Hock, M. Stuber, "Evaluation of a fall detector based on accelerometers: a pilot study," *Medical & Biological Engineering & Computing*, vol. 43, pp. 1146−1154, 2005. Article (CrossRef Link)
[7] T. Degen, H. Jaeckel, M. Rufer, S. Wyss, "SPEEDY: A Fall Detector in a Wrist Watch," in *Proc. of 7th International Symposium on Wearable Computers*, 2003. Article (CrossRef Link)
[8] B. Mattia, R. Leopoldo, "Wrist-Worn Fall Detection Device - Development and Preliminary Evaluation," *BIODEVICES 2009*, pp. 368-371, 2009.
[9] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, "Classification and regression trees," Wadsworth & Brooks/Cole Advanced Books & Software, 1998.
[10] F.van der heijden, R.P.W. Duin, D.de Ridder, "Classification, Parameter Estimation and State Estimation," Wiley, 2004.
[11] J.H. Yo, M.S. Nixon, "Automated Markerless Analysis of Human Gait Motion for Recognition and Classification," *ETRI Journal*, vol. 33, no. 2, 2011. Article (CrossRef Link)

Chankyu Park received the MS degree in electronics engineering from Kyungpook National University in 1997, the MS degree in software engineering from Carnegie-Mellon University in 2005, is currently a Ph.D candidate in electric & electronics engineering from KAIST. He joined Electronics and Telecommunications Research Institute (ETRI) in 1997 and has been a Senior Member of Engineering Staff in the Intelligent Robot Research Division since 2004. His research interests are in pattern recognition, machine learning, computer vision and health-care robotics.

Jaehong Kim received his PhD from Kyungpook National University in 1996. He is a research scientist at ETRI. He is interested in the elderly-care aspects of human-robot interaction.

Joo-Chan Sohn received the MS degree in management information systems from HankookUniversity of Foreign Studies in 1990. Since joining ETRI, he has been involved with electronic commerce systems and intelligent e-business systems. Currently, he is a principal member and director of Engineering Staff in the Robot/Cognitive System Research Department at ETRI. His researchinterests are service infrastructure for robots and robotic system for elderly and smart defense.

Ho-Jin Choi is a faculty member of the Dept. of Computer Science at Korea Advanced Institute of Science and Technology(KAIST), Daejeon, Korea. In 1982, he received a BS in Computer Engineering from Seoul National University, Korea, in 1985, an MSc in Computing Software and Systems Design from Newcastle University, UK, and in 1995, a PhD in Artificial Intelligence from Imperial College, London, UK. From 1982 to 1989, he worked as a senior engineer for DACOM Laboratory, Korea, and between 1995 and 1996, as a post-doctoral researcher at IC-PARC, Imperial College, London. From 1997 to 2002, he was a faculty member at Korea Aerospace University, then from 2002 to 2009, a faculty member at Information and Communications University(ICU), Korea, and since 2009 he has been with the Dept. of Computer Science at KAIST. Between 2002 and 2003, he visited Carnegie Mellon University(CMU), USA, and became an adjunct professor of CMU for the program of Master of Software Engineering(MSE). Between 2006 and 2008, he served as the Director of Institute for IT Gifted Youth at ICU. His research interests include artificial intelligence, data mining, software engineering, and biomedical informatics.