ML Project : Machine Learning Methods Applied to Human Physical Activity Classification Using On-Body Sensors

Mohamed-El-Amine Seddik Marwen Sallem Kais Slimi Oussama Bouraoui

January 26, 2017

Abstract

This paper describes our choice for the machine learning project subject, we chose to work on applying machine learning methods to the problem of human physical activity classification, based on data that comes from on-body sensors. Basically, the aim of this project is to explore different techniques of machine learning and to apply them to this particular problem.

Motivation

The ability of designing a system that automatically classifies the physical activity performed by a given person is very attractive for many applications in various fields such that health-care monitoring and in developing smart human-machine interfaces. There are several techniques to handle this task, and mainly some of them are based on computer vision where some others aims to use on-body sensors. Based on [?] we chose to work on data that comes from on-body sensors. In fact, there is many datasets available on the internet that are used for this special problem, we provide below a link to a dataset that we plan to use for our project.

Goal

As illustrated by the title, our main goal is to classify the Human Physical Activity. To reach this goal we plan to explore different machine learning methods (Probabilistic based methods like HMM and geometric based ones as SVM k-NN ...) and to apply them to this particular task, at the end we aim to provide the performances of the different applied methods.

Dataset

- https://archive.ics.uci.edu/ml/datasets/Activity+Recognition+from+Single+Chest-Mounted+Accelerometer

Introduction

Several techniques have been used to recognize Human activities such as video-based sensors, wearable-based sensors, environmental sensors and object sensors like smart phones. In this paper, wearable-sensors based systems for activity recognition has been used.

Human activity recognition or (HAR), begins by collecting the raw data from sensors worn on certain parts of the body. The sensors will provide different data depending on the part of the body where it's worn (for example wrist, chest, ...) and the technical characteristics of the sensor itself. One of the biggest challenges of HAR is to overcome this variability in sensor characteristics. The sensors usually have limited processing power which makes it difficult to get a high frequency sampling or do any kind of preprocessing on the sensors.[?]

We are therefore unable to use the raw data directly for the classification task. We need to do several preprocessing steps depending on the classification method we want to use. In section 1 we will explain the state of the art feature extraction techniques encountred in such problems.

Another issue in HAR is Intraclass variability: Different people will do the same activity differently. Their data on the sensors will not be very similar but it will have the same label (in the supervised setting) or the classifier has to assign them the same label (in the unsupervised setting). This can be overcomed by collecting more data or develop person-independant features that are more robust to this variability. In the section 2, we will present a supervised technique that aims to classsify the human activities using LDA and KDA classifier. But generally the supervised approaches require a large amount of labelled data. They are too expensive. For that, one has to rely on unsupervised approaches. Hence, we will present in 3 an unsupervized technique based upon joint segmentation of multidimensional time series using a Hidden Markov Model (HMM) in a multiple regression context and is denoted by MHMMR: Multiple Hidden Markov Model Regression.

1 Pre-processing and features extraction

Acclerometers provide three axis time series A_x, A_y and A_z. Once sampling on a known frequency, we obtain a different signal for each class, for instance the figure 1

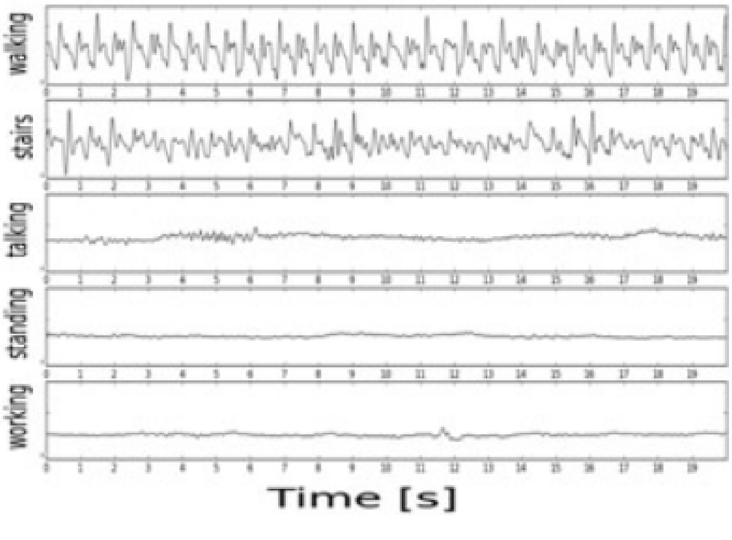


Figure 1: Accelerometer Data for Five Different Activities [?]

To be able to apply Machine Learning techniques, we have to extract features from these given signals. These latter is mainly done on two different phases :

In the first phase we are going to widen our signal databases by adding new signals :

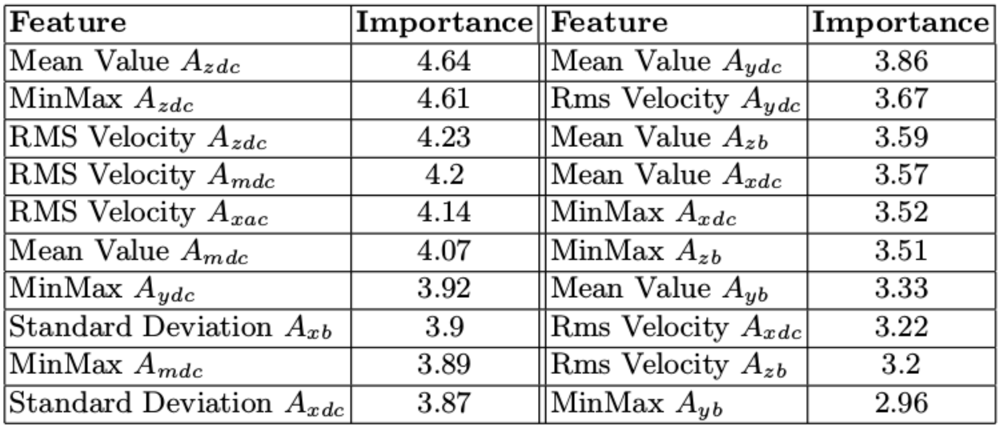
- We will compute the magnitude of the acceleration : $A_m = \sqrt{A_x^2 + A_y^2 + A_z^2}$ [?].

- We will separate every axis time-series in high-frequency (AC) components that captures dynamic motion, and low-frequency (DC) components related to the gravity's influence that captures static postures. Thereby, we will obtain for each time series three additional time series that corresponds to the signal without filtering, the signal after applying a high-pass filter and the signal after applying a low-pass filter.

Now we have 12 signals extracted from the three initial signals A_x, A_y and A_z. Having these new set of signals, we will split them into some windows in order to compute features. In fact, we apply a windowing and overlapping process which means that we will take windows of 52 sample (1 second of data) with 50% of overlapping between windows. Thus, we can move to the second phase, i.e extracting features :

- We compute the RMS of the velocity (integration of acceleration) and mean value of Minmax sums[1] . In [?] we showed that theses features are the most important for the proposed Random Forest classifier in the Table 1.
- Basic statistics as mean value, standard deviation, skewness, kurtosis, correlation between each pairwise of accelerometer axis. In fact, Manini and Sabatini shows in [?] that these features proved to be useful in HAR.
- Other features like ARIMA coefficients [?] and energy of the coefficients of some level wavelet decomposition [?] are also of great importance.

Table 1. List of Features selected by Random Forest



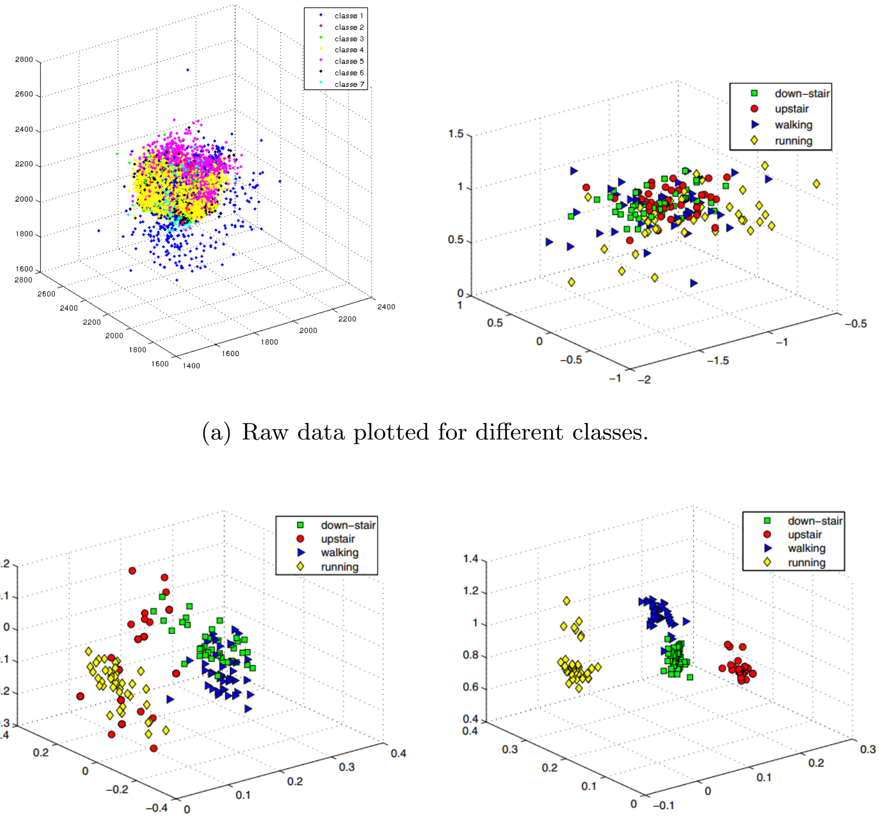
Feature	Importance	Feature	Importance
Mean Value A_{zdc}	4.64	Mean Value A_{ydc}	3.86
MinMax A_{zdc}	4.61	Rms Velocity A_{ydc}	3.67
RMS Velocity A_{zdc}	4.23	Mean Value A_{zb}	3.59
RMS Velocity A_{mdc}	4.2	Mean Value A_{xdc}	3.57
RMS Velocity A_{xac}	4.14	MinMax A_{xdc}	3.52
Mean Value A_{mdc}	4.07	MinMax A_{zb}	3.51
MinMax A_{ydc}	3.92	Mean Value A_{yb}	3.33
Standard Deviation A_{xb}	3.9	Rms Velocity A_{xdc}	3.22
MinMax A_{mdc}	3.89	Rms Velocity A_{zb}	3.2
Standard Deviation A_{xdc}	3.87	MinMax A_{yb}	2.96

Now, we will have more than 300 features. It's possible to use them either in Single-Frame approaches as " Linear and Kernel Discriminant Analysis approaches " or in Sequential approaches like " Hidden Markov Model Regression " which we will describe in the next two sections.

2 Linear and Kernel Discriminant Analysis approaches

One observes the raw data of the chosen dataset, we acknowledge an overlapping between the different classes (see Fig 2(a)), this overlapping is due to the high with-in and low between-class variances. Based on [?], in order to fix this overlapping problem, we present in this section the LDA and KDA models, which are a supervised classification approaches that utilizes the class specific information maximizing the ratio of the within and between class scatter information. In particular, the KDA is a generalization of LDA applied in a RKHS feature space.

[1] The Minmax sums are computed as the sum of all the differences of the ordered pairs of the peaks of the time series.[?]



(a) Raw data plotted for different classes.

(b) Results of applying LDA (left) and KDA (right) to the raw data.

Figure 2: Visualization of the different classes before and after applying Linear and Kernel Discriminant Analysis, we notice that the classes are perfectly separable ones applying the KDA (see 2(b) right).

2.1 Theory about Linear Discriminant Analysis

The LDA responds to the question of how do we use the label information in finding informative projections of our data? To that purpose it considers maximizing the following objective:

$$(\mathcal{P}): \quad \max_{\omega} J(\omega) = \max_{\omega} \frac{\omega^T S_B \omega}{\omega^T S_W \omega} \tag{1}$$

Where S_B is the between classes scatter matrix and S_W is the within classes scatter matrix. And are given by:

$$S_B = \sum_c (\mu_c - \bar{x})(\mu_c - \bar{x})^T \quad S_W = \sum_c \sum_{i \in c} (x_i - \mu_c)(x_i - \mu_c)^T \tag{2}$$

Where $\bar{x}$ overall mean of the data.
J being invariant with respect to re-scaling of the vector $\omega \rightarrow \alpha\omega$. Thus we can always choose ω such that $\omega^T S_W \omega = 1$, so that the problem $(\mathcal{P})$ could be written as:

$$(\mathcal{P}): \quad \min_{\omega} -\frac{1}{2}\omega^T S_B \omega \ \texttt{s.t.} \quad \omega^T S_W \omega = 1 \tag{3}$$

The Laplacian of $(\mathcal{P})$ is thus:

$$\mathcal{L}_{\mathcal{P}}(\omega, \lambda) = -\frac{1}{2}\omega^T S_B \omega + \frac{1}{2}\lambda(\omega^T S_W \omega - 1) \tag{4}$$

Finally, taking the derivative with respect to ω of the Laplacian, one get:

$$S_B\omega = \lambda S_W\omega \Rightarrow S_W^{-1}S_B\omega = \lambda\omega \tag{5}$$

S_B is symmetric positive definite (p.d.), thus $S_B = S_B^{1/2}S_B^{1/2}$ where $S_B^{1/2} = U\Lambda^{1/2}U^T$ (where $U\Lambda U^T$ is the eigen-decomposition of S_B). So performing the change of variable $v = S_B^{1/2}\omega$, one get:

$$\underbrace{S_B^{1/2}S_W^{-1}S_B^{1/2}}_{\texttt{symmetric p.d.}} v = \lambda v \tag{6}$$

Finally, the goal of the LDA is to find the eigenvector corresponding to the largest eigenvalue of the symmetric p.d. matrix $S_B^{1/2}S_W^{-1}S_B^{1/2}$. Denote this vector v^*, thus $w^* = S_B^{-1/2}v^*$.

2.2 Theory about Kernel Discriminant Analysis

To kernelize the LDA, we consider the problem in a feature space $\mathcal{F}$ induced by some non-linear mapping function $\phi : \mathbb{R}^d \to \mathcal{F}$. An inner product $< .|. >_{\mathcal{F}}$ can be defined in $\mathcal{F}$ which makes for so called reproducing the kernel Hilbert Space (RKHS). More precisely $< \phi(x_i)|\phi(x_j) >_{\mathcal{F}} = K(x_i, x_j)$ holds where $K(.,.)$ is a p.s.d. kernel function. Thus, to find the linear discriminant in $\mathcal{F}$, we need to maximize:

$$(\mathcal{P}_K): \quad \max_{\alpha} J_\phi(\alpha) = \max_{\alpha} \frac{\alpha^T S_b^\phi \alpha}{\alpha^T S_w^\phi \alpha} \tag{7}$$

Where

$$\begin{aligned} S_b^\phi &= \sum_c n_c(K_cK_c^T - KK^T) \\ S_w^\phi &= K^2 - \sum_c n_cK_cK_c^T \\ K_c = \frac{1}{n_c}\sum_{i\in c} K_{i,j}, &\quad K = \frac{1}{n}\sum_i K_{i,j} \end{aligned} \tag{8}$$

According to the equation (6), one get the α which maximizes the objective above, denote by α^* this solution. Thus, ω_* in the initial space is given by:

$$\omega_* = \sum_i \alpha_i^*\phi(x_i) \tag{9}$$

And finally using the kernel trick, we don't need to compute explicitly ϕ, and the projection of a new sample x over ω_* is given by:

$$\omega_*^T\phi(x) = \sum_i \alpha_i^*K(x_i, x) \tag{10}$$

3 An Unsupervised Approach for Automatic Activity Recognition based on Hidden Markov Model Regression [?]

In the proposed model, each activity is represented by a regression model and the switching from one activity to another is governed by a hidden Markov chain. The MHMMR parameters are learned in an unsupervised way from unlabelled raw acceleration data acquired during human activities using the Expectation-Maximization (EM) algorithm where no activity labels are needed.

The acceleration data are presented as multidimensional time series presenting various regime changes. In such context, the goal is to provide an automatic partition of the data into different segments (regimes),each segment being considered afterwards as an activity.

In Hidden Markov Model Regression (HMMR), each time series is represented as a sequence of observed univariate variables $(y_1, y_2, \ldots, y_n)$, where the observation y_i at time t_i is assumed to be generated by the following regression model:

$$y_i = \beta_{z_i} + \sigma_{z_t}\epsilon_i \; ; \; \epsilon_i \sim \mathcal{N}(0,1) \; , \; (i = 1, \ldots, n) \tag{11}$$

where $z_i \in (1, \ldots, K)$ is a hidden discrete-valued variable. In this application case, z_i represents the hidden class label (activity) of each acceleration data point and K corresponds to the number of considered activities. The variable z_i controls the switching from one polynomial regression model associated to one activity, to another of K models at time t_i.

The HMMR assumes that the hidden sequence $z = (z_1, \ldots, z_n)$ is a homogeneous Markov chain of first order parameterized by the initial state distribution π and the transition matrix A. Regarding the multiple regression case, the model can be formulated as follows:

$$\begin{aligned}
y_i^{(1)} &= \beta_{z_i}^{(1)T} + \sigma_{z_t}^{(1)}\epsilon_i && (12)\\
y_i^{(2)} &= \beta_{z_i}^{(2)T} + \sigma_{z_t}^{(2)}\epsilon_i && (13)\\
\vdots \quad & \quad \vdots && (14)\\
y_i^{(d)} &= \beta_{z_i}^{(d)T} + \sigma_{z_t}^{(d)}\epsilon_i && (15)\\
& && (16)
\end{aligned}$$

where d represents the dimension of the time series.

The Multiple HMMR model is therefore fully parameterized by the parameter vector :

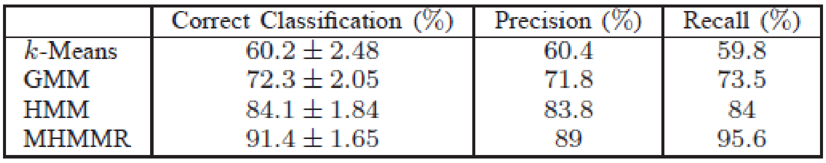
$$\theta = (\pi, A, B_1, \ldots, B_k, \Sigma_1, \ldots, \Sigma_k) \tag{17}$$

where B_k is the matrix of the multiple regression model parameters associated with the regime (class) $z_i = k$ and Σ its corresponding covariance matrix.

The parameters are estimated by maximizing the observed data likelihood through the Expectation-Maximization (EM) algorithm. The log-likelihood to be maximized in this case is written as follows:

$$\mathcal{L}(\theta) = \log p(y_1, \ldots, y_n; \theta) = \log \sum_{z} p(z_1; \pi) \prod_{i=2}^{n} p(z_i | z_{i-1}; A) \prod_{i=1}^{n} \mathcal{N}(y_i; B_{z_i}^T, t_i, \Sigma_{z_i}) \tag{18}$$

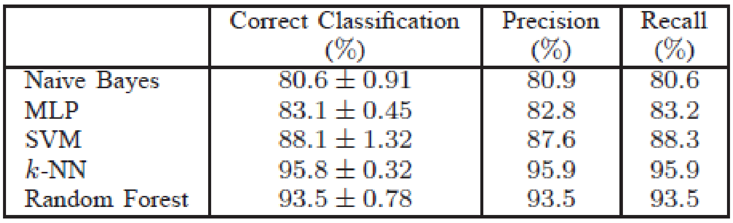
To evaluate the performence of this method, it has to be compared to other algorithms. Compared to standard unsupervised classifiers, the proposed MHMMR outperforms them since it provides the best classification rate, 91.4%.



	Correct Classification (%)	Precision (%)	Recall (%)
k-Means	60.2 ± 2.48	60.4	59.8
GMM	72.3 ± 2.05	71.8	73.5
HMM	84.1 ± 1.84	83.8	84
MHMMR	91.4 ± 1.65	89	95.6

Comparison of the performance in terms of Correct Classification, Recall and Precision of the four unsupervised classifiers

For the supervised learning, the k-NN ($k = 1$) gives the highest classification rates with 95.8%, followed by the Random Forest with 93.5%.



	Correct Classification (%)	Precision (%)	Recall (%)
Naive Bayes	80.6 ± 0.91	80.9	80.6
MLP	83.1 ± 0.45	82.8	83.2
SVM	88.1 ± 1.32	87.6	88.3
k-NN	95.8 ± 0.32	95.9	95.9
Random Forest	93.5 ± 0.78	93.5	93.5

Comparison of the performance in terms of Correct Classification, Recall and Precision of the five supervised classifiers