



Human-centric algebraic machine learning

Sensor based context and activity recognition methods based on AML

Deliverable D5.1

WP5 - Activity recognition with AML



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 952091.

Consortium



ALGEBRAIC AI



Chantalimaud
Foundation



German
Research Center
for Artificial
Intelligence



TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN



Project

Title: ALMA: Human Centric Algebraic Machine Learning
 Acronym: ALMA
 Coordinator: Proyectos y Sistemas de Mantenimiento SL (eProsima)
 Grant agreement: 952091
 Call: H2020-EIC-FETPROACT-2019
 Program: Horizon 2020
 Start: 01 September 2020
 Duration: 48 months
 Website: <https://alma-ai.eu>
 E-mail: alma@eprosima.com
 Consortium: **Proyectos y Sistemas de Mantenimiento SL (eProsima) (EPROS)**, Spain
Champalimaud Foundation (CF), Portugal
ALGEBRAIC AI (ALG), Spain
German Research Center for Artificial Intelligence (DFKI), Germany
Inria (Inria), France
TU Kaiserslautern (TUK), Germany
FIWARE Foundation e.V. (FF), Germany
Technical Research Centre of Finland (VTT), Finland
Carlos III University (UC3M), Spain

Deliverable

Number: **D5.1**
 Title: **Sensor based context and activity recognition methods based on AML**
 Month: 9
 Work Package: WP5 - Activity recognition with AML
 Work Package leader: DFKI
 Deliverable leader: Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
 Deliverable type: DEM: Code and Evaluation Report
 Dissemination level: Confidential, only for members of the consortium (including the Commission Services) (CO)
 Date of submission: 2021-05-31
 Version: v1.0
 Status: Not finished

Version history

Version	Date	Responsible	Author/Reviewer	Comments
v1.0	2021-04-07	DFKI	Matthias Tschöpe	First Draft
v1.1	2021-04-25	ALG	F. Martin-Maroto	Embedding and results
v1.2	2021-04-07	DFKI	Vitor Fortes Rey, Paul Lukowicz and Gonzalo G. de Polavieja	Revision

Executive summary

The goal of this workpackage is to explore the potential advantages of AML (ability to naturally combine data and prior knowledge, less impact of unbalance data, no or very small hyperparameter search, see [1]) to test whether it is possible to either improve the accuracy of human activity recognition methods and/or improve the interpretation of the resulting models.

This is the first of three deliverables to achieve this goal. In this first deliverable, we study how AML can be applied to standard activity recognition tasks and data sets. This is a starting point for developing an understanding of how to use formal knowledge of the human activity problem, along with the data, to obtain AML models. As a first step we demonstrate how to use formal knowledge to embed the notion of intensity value of sensor data using order relationships and that sensor readings are independent.

We performed tests to show that, for this embedding, we can already obtain models with a generalization performance comparable to standard methods, but still not quite as good as state-of-the-art Deep Learning approaches.

These results encourage two lines of research for the next deliverables. On one hand, we will work to find embeddings that are able to better encode expert knowledge about the recognition problem. It is conceivable that better embeddings and improvements in pre-processing of the data improve upon state-of-the-art methods, especially for the case in which we do not have validation dataset and the training is very unbalanced. In addition to accuracy results, we will study advantages in deployment times. Also, we will study the atoms of the model to gain insight in what is being learned in the AML approach.

Contents

Executive Summary	3
Contents	4
Acronyms	5
1 Introduction	6
1.1 Data used	6
1.2 Feature Computation using a Windowing Approach	7
1.3 Discretizing the Features	10
2 How to embed the human activity problem into AML	10
2.1 A naive embedding with no formal knowledge	10
2.2 Formal knowledge to express the notion of intensity	11
2.3 Formal knowledge to express the notion of independency	12
2.4 Training relations for data	12
3 Training details	12
4 Determining the class to which a test example belongs	13
4.1 Using a validation dataset	13
4.2 Without validation	14
5 Classification Results	16
5.1 Metrics	16
5.2 Overall Results	16
5.3 Results without using a validation set	18
6 Conclusion	19
7 Appendix	21
References	27

Acronyms

1 NN	1-Nearest Neighbours.
3 NN	3-Nearest Neighbours.
ADL	Activities of Daily Living.
AML	Algebraic Machine Learning.
DT	Decision tree.
FN	False Negative.
FP	False Positive.
LDA	Linear Discriminant Analysis.
MLP	Multilayer Perceptron.
NCC	Nearest Centroid Classifier.
QDA	Quadratic Discriminant Analysis.
TN	True Negative.
TP	True Positive.

1 Introduction

Human Activity recognition (HAR) is an established research topic at the interface of artificial intelligence (AI) and Ubiquitous Computing [2]. Today wearable and environmental sensors are becoming virtually omnipresent. There has been the expectation that as large amounts of data from such omnipresent sensors become available HAR will be able to benefit from the advances in Deep Learning techniques in the same way as fields such as computer vision and speech recognition. However, while the application of deep learning methods to HAR has produced undeniable advances [3] the progress has so far been far less significant than in computer vision or NLP. Thus in computer vision deep learning techniques became popular in the last decade [4] and till today have managed to performance that is comparable or even superior to humans [5] with double digits improvements in the recognition rates. By contrast in sensor based activity recognition the improvement over the last decade was mostly in single digits with the recognition of complex activities still often being significantly below the requirement of many applications.

In a nutshell the above is due to high complexity, dynamism, and variability of real world environments and human actions performed in such environment combined with the difficulty of acquiring large amounts of **labeled** training data. A promising solution to that problem is the inclusion of human expert knowledge and physical constraints in the training process. As explained in the proposal we expect that AML, which poses the learning task in form of algebraic expressions, can be an effective tool for this purpose. On the other hand at this stage it is unclear how to actually use AML for HAR problems and what sort of baseline performance to expect. Answering these initial questions is the aim of this deliverable. In other words we want to develop the basic techniques for using AML to learn a solution to activity recognition tasks and evaluate how well an initial, simple AML implementation performs on a standard problem. In particular the question is how to best embed real valued highly dimensional sensor input spaces in discrete algebraic structures without losing information or producing excessive computational loads.

To achieve the above we focus on a simple well researched "toy" problem (basic modes of locomotion) on a popular benchmark data set. One big advantage of the modes of locomotion problem is that while it is today still routinely used problem on testing complex Deep Models on large dimensional feature spaces it can also be easily solved by using physical constraints on few sensor signals selected through expert knowledge. For example, walking is the only class with a significant energy in the 1-3Hz range of the norm of virtually any sensor position. Sitting standing and laying down can be separated by looking at the angle of the gravity vector towards the lower leg, upper leg and torso sensors. This is expert knowledge that we will try to encode in AML in the next project phase before proceeding to more complex activities.

1.1 Data used

Based on the above considerations, for the investigation described in this deliverable we use the OP-PORTUNITY dataset [6]. It is well studied with the results of many different types of evaluation and recognition approaches being thoroughly documented [7]. In addition to the basic modes of locomotion task (standing, walking, sitting, lying) the data set has a variety of more complex and subtle activities such as "Breakfast", "Cleaning up" and "Going back to sleep". These activities consist of many sub-activities also called gestures. For example, Breakfast consists of "open the door 1", "close the door 1", "open the door 2", "close the door 2", "reach item", "move item" and so on [6]. Detecting those gestures will be part of our next deliverable. At this stage of our work we focus only on the current mode of locomotion. Distinguishing more complex gestures individually will be addressed in the next deliverable where we will also explore more sophisticated ways of embedding expert knowledge in AML.

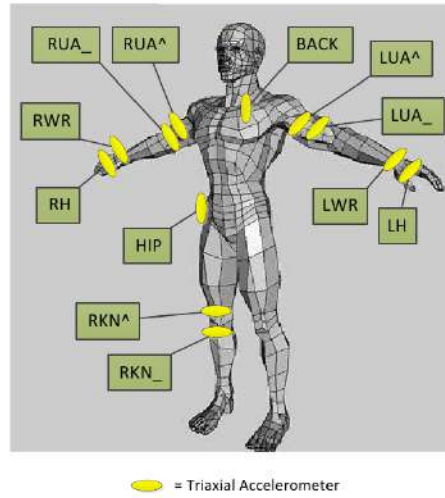


Figure 1: The yellow ovals visualize the positions of the accelerometers. The image is from Ricardo Chavarriaga et al. [8]

In general the OPPORTUNITY dataset provides a lot of on-body, ambient and object sensors. Since we focus on locomotion activities we can restrict the set of used sensors to 12 acceleration sensors. Each acceleration sensor measures the acceleration in three directions. Figure 1 shows the positions of the acceleration sensor.

1.2 Feature Computation using a Windowing Approach

Multimodal, sensor based HAR systems typically pre-process the raw sensor values by computing a set of standard features both in the time and in the frequency domain on a sliding window. When using Deep Learning approaches such features are often packaged as a 2-D "pseudo-image" before being fed into the network which allows the use of architectures from image processing. Sometimes the raw features are also used, although they need to be at least normalized and the different sampling rates need to be addressed.

Figure 2a shows a raw signal for one acceleration sensor in all three directions. The frequency of all sensors is 30 Hz. Thus, the sensor measures the current acceleration every 33.33 milliseconds and uses it to create a new sample.

A window is a set of consecutive samples of a raw signal, see the orange parts in Figure 2b and Figure 2c. To create a window, we first have to define a window size. The size of a window depends on the activities. More precisely, on the duration of a repeating action of an activity. For example, considering the activity walking, the duration of making one or two steps defines the window size. In 2018, Frédéric Li et al. [9] compared different approaches with several window sizes on the OPPORTUNITY data set. The considered window sizes were between 1 and 3 seconds. Since their accuracy was higher for larger window sizes, we also used a window size of 3 seconds. Together with the sampling rate of 30 Hz, we get a window with 90 samples. To create the next window, we have to define a new start and end time. We decided to have a 25% window size of overlap. Hence, the next window starts (and also ends) 22 samples later as the window before. Each window can be represented as a matrix of size 90×37 . As mentioned before, 90 corresponds to the number of samples. Each sample consists of 36 sensor signals (12 sensors, each of them measures the acceleration of 3 directions) and one label.

Before we compute the features for each signal in each window, we add another "signal". The advantage

of this signal is the independence of the direction. This fourth signal is the magnitude of all three raw signals. Let w be a window and w_r be the r -th row of this window. From above, we know $1 \leq r \leq 90$. Further, let $\sigma(s, d, w_r)$ be the raw signal for sensor s , direction d , in w_r . The magnitude $m(s, w_r)$ for s in w_r , is defined as:

$$m(s, w_r) = \sqrt{\sigma(s, x, w_r)^2 + \sigma(s, y, w_r)^2 + \sigma(s, z, w_r)^2} \quad (1)$$

To compute the fourth signal, we have to compute the magnitude for each sensor s in the whole window w . Hence, we have:

$$mag(s, w) = (m(s, w_1), m(s, w_2), \dots, m(s, w_{90})) \quad (2)$$

Thus, the window size increases to 90×49 . This is because now we have 4 “directions” for all 12 sensors, plus the label column. Finally, we compute the features for each of the 48 feature columns. Again, we used the work of Li et al. [9] as baseline of feature selection. We decided to compute the following features:

- average
- standard deviation
- minimum
- maximum
- median
- kurtosis
- skew
- zero crossings

For each of the 48 signal-columns in each window, we compute the above 8 features. This yields to a sample with 384 features. Unfortunately in 37.5% of the sessions, the sensor *LH* (*Left Hand*), see Figure 1

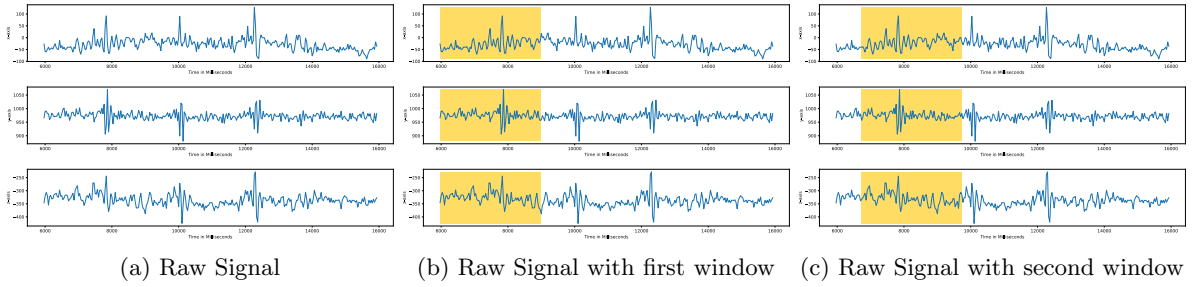


Figure 2: Each subfigure shows a small snippet of a raw signal (all three directions) for the same sensor (RKN). Blue: The raw signal, given in the OPPORTUNITY dataset. Orange: A window.

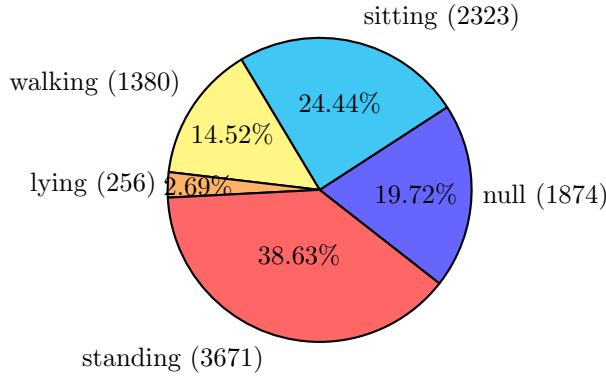


Figure 3: Training data distribution after computing the features.

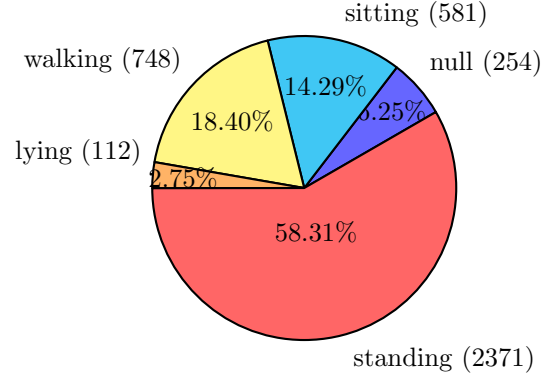


Figure 4: Validation data distribution after computing the features.

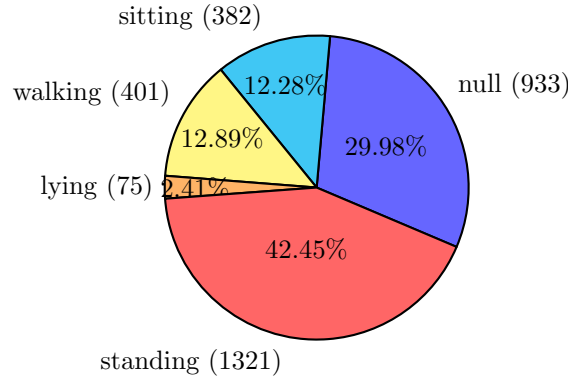


Figure 5: Test data distribution after computing the features.

returned NaN values, which means there was some problem with the sensor at that time. Therefore we have not used this sensor, and consequently our samples consist of $11 \cdot 4 \cdot 8 = 352$ features. We can again represent these data as a matrix, where each sample is represented as a row, the first 352 columns correspond to the features and the last column corresponds to the labels. This matrix is often called a Feature-Matrix.

During the creation of the windows, we check two things. First, if there is a NaN value in the window, then we discard this window. Second, only if more than 75% of the labels are the same, then we take this window, otherwise we discard it.

The OPPORTUNITY dataset contains six recordings for each of the four participants, i.e. in total 24 recording files respectively [6, 10]. Five of these are Activities of Daily Living (ADL) recordings and one is a drill recording [6, 8, 10]. For training we used all recordings of subject one and two, for validation we used all recordings of subject three and as test set we used all recordings of subject four. This gives us a training set with 9504 samples, a validation set with 4066 samples, and a test set with 3112 samples. This is also known as leave-person-out [11] and is a harder task than leaving sessions out, as the model needs to generalise across subjects. Another challenge in Activity Recognition is unbalanced data. Figure 3 and Figure 5 show the data distribution after computing the features for the training and test data. As we can see there, the data is highly unbalanced, with the “lying” class with under 3% of instances.

1.3 Discretizing the Features

AML is based on Discrete Mathematics, so we need a strategy to discretize the data without losing much information. In this chapter we will explain two methods to discretize the data. Both methods describe a mapping from a continuous value to the corresponding discrete value. To compute this method, we only use the train-val Feature-Matrix M , then we used this function to discretize the values in the test or validation Feature-Matrix. For further explanations we define $M_{i,\cdot}$ as the i -th row and $M_{\cdot,j}$ as the j -th column.

Our first idea was to compute a histogram for each column in M . A histogram distributes the range of the values in k equally distributed areas, also called bins. This can already be a problem, as some parts of the distribution may need more detail than others. For example, if there are two frequently occurring values that are close to each other, then they could be in the same bin. But since they are frequently occurring, it might be better, to have to different bins, because both values might be important to correctly classify the activity. So we quickly moved on to a better solution before implementing this idea.

The second idea was to use a k -means algorithm, to find k important (i.e. close to frequently occurring) values. Here we have another problem: The cluster values found by the k -means algorithm still might be continous. To formally solve this problem we need some definitions. Let C_i be the i -th cluster with $1 \leq i \leq k$, then by $v(C_i)$ we denote the cluster value of C_i , and by $n(C_i)$ the cluster number (initially unkown) of C_i . Further, let L be a list of all C_i , the order of the C_i in L does not matter. To compute the cluster numbers $n(C_i)$, we first sort all C_i in L in increasing order of $v(C_i)$. Now we set $n(C_i)$ to the list index of C_i in L . Since the indices are integers, we can use $n(C_i)$ as a discrete representation of C_i respectively $v(C_i)$. Since, AML was not very optimised at this time, we could not use too many clusters. Therefore we decided to use $k = 16$ clusters. Figure 6 visualizes the above idea.

2 How to embed the human activity problem into AML

One of the advantages of AML is that data and formal knowledge are treated identically. For human activity recognition, we could use a variety of embeddings with different formal knowledge, and for this first deliverable we have formalized the notion of sensor intensity and the notion of independency of readings.

2.1 A naive embedding with no formal knowledge

The problem input consists of the feature readings from the sensor channels (including the calculated norms) calculated for each window, generating 352 numbers per window, each one discretized into 16 values. We will also use 6 constants to represent the 6 output classes.

A very naive embedding of the problem would express each training example as the idempotent summation of 352 out of the 352×16 possible constants. Let $r(s)$ be a vector of the actual sensor readings for a given training example, with each position in the vector one of the possible 16 readings and the length of the vector 352, with $s = 0, \dots, 351$. Let the output value k represent the kind of physical activity performed. For each training example a term T was created representing the problem input:

$$T = r(0) \odot r(1) \odot \dots \odot r(351). \quad (3)$$

Using 6 constants O_k with $k = 0, \dots, 5$ to represent the output activity classes, we can embed each positive training example as

$$O_k \leq T. \quad (4)$$

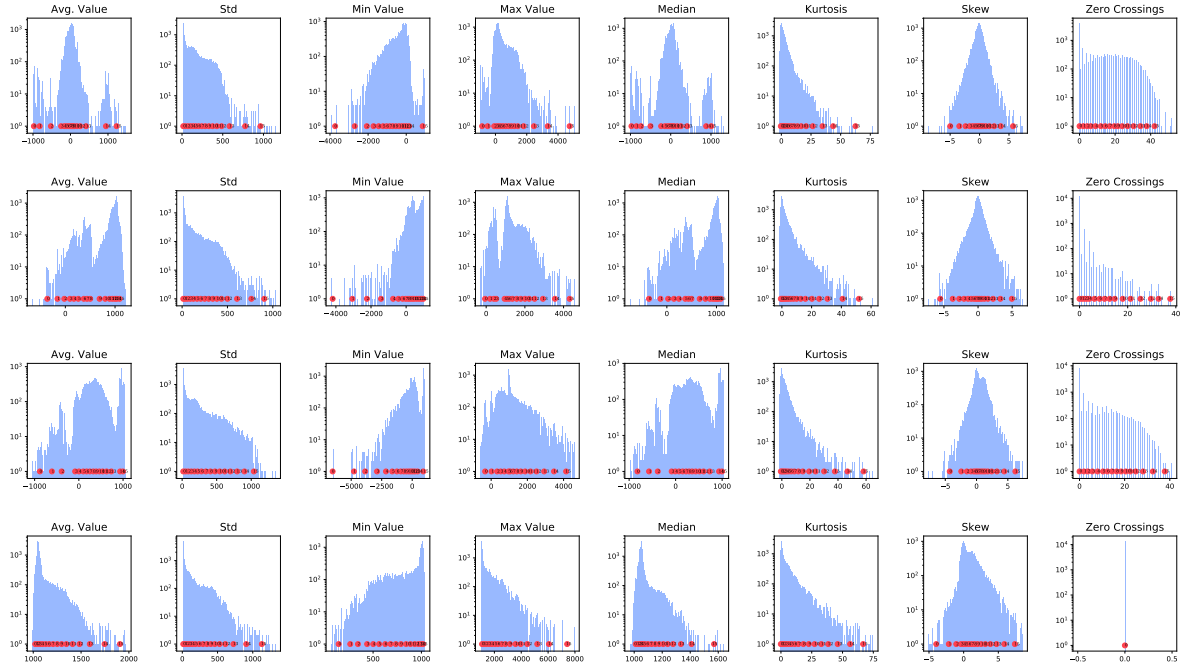


Figure 6: This Figure shows four rows and eight columns. Each row represents a sensor "direction", x, y, z and magnitude (top-down). Each column represents a feature for this direction. For a better visualization we show each subplot shows a histogram for each feature. Here we used 100 bins (much more as Algebraic Machine Learning (AML) can handle). The y -axis how often those values occur. The red dots in the bottom of each subplot represents the cluster position, computed on the feature values. Each red dot has the corresponding cluster number $n(C_i)$. We used $k = 16$ clusters.

2.2 Formal knowledge to express the notion of intensity

AML can use the same type of order relations to not only discuss data as in the previous section, but also express any formal knowledge we might have about the problem. In our case we know that a sensor of value 4 has a higher value than a sensory of value 3 and this 3 value is higher than 2. The naive embedding of the previous section did not use this. To encode the notion of intensity, we define new constants and new relations.

We define for each feature $30 = 2 \times (16 - 1)$ constants separated into two sets of 15 constants each, one set for "lower or equal" constants and another set for "greater than" constants. A total of $352 \times 30 = 10,560$ constants are used to describe the input and another 6 constants are used for the output, which represents the activity being performed.

For each sensor s , the 30 constants associated to the sensor are $le_{0,s}, le_{1,s}, \dots, le_{14,s}$ and $g_{0,s}, g_{1,s}, \dots, g_{14,s}$. The formal knowledge we want to express is the notion of intensity, which we will represent as ordered sets

$$le_{i+1,s} \leq le_{i,s} \quad (5)$$

$$le_{i,s} \not\leq le_{i+1,s} \quad (6)$$

$$g_{i,s} \leq g_{i+1,s} \quad (7)$$

$$g_{i+1,s} \not\leq g_{i,s} \quad (8)$$

forming one ascending chain and one descending chain for each sensor. The goal is embedding the discretized sensor reading which corresponds to an intensity, i.e a numerical value and, therefore, represented as an ordered set in a discrete system.

2.3 Formal knowledge to express the notion of independency

The constants associated to every sensor are input constants with independent readings. This independence of the input constants was embedded with the order relations

$$le_{i,s} \not\leq \odot_{s' \neq s} \odot_{i'} (le_{i',s'} \odot g_{i',s'}) \quad (9)$$

$$g_{i,s} \not\leq \odot_{s' \neq s} \odot_{i'} (le_{i',s'} \odot g_{i',s'}), \quad (10)$$

where the subindex in the operator $\odot$ indicates an idempotent summation for all values of the subindex.

2.4 Training relations for data

Each training example can be represented as a vector $r(s)$ corresponding to the $s = 0, \dots, 351$ discretized sensor readings and one output value k representing the kind of physical activity performed. For each training example, a term T was created representing the problem input:

$$T = \odot_s \left\{ \begin{array}{ll} le_{0,s} & r(s) = 0 \\ le_{r(s),s} \odot g_{r(s)-1,s} & 0 < r(s) < 15 \\ g_{14,s} & r(s) = 15 \end{array} \right\} \quad (11)$$

and the training example was embedded with the positive relation

$$O_k \leq T \quad (12)$$

and $4 = 5 - 1$ negative relations expressing that when some data belongs to a class it does not belong to any other class,

$$O_{k' \neq k} \not\leq T. \quad (13)$$

The complete theory we have used to embed the human activity problem then consists on how training data is labelled, **Eqs. 12** and **Eqs. 13**, and the formal knowledge relations **Eqs. 5** and **Eqs. 9**. We used the AML description language to build these relations easily and to input them into the computation of the model. See AMLDL 0.1[12] for a description of the language and **Appendix** for the code of this embedding.

3 Training details

Since the number of input parameters was quite large, in order to train a model using the available hardware, a i9 CPU with 28 cores, each output class was trained independently and consecutively. As a result, five models M_0, M_1, M_2, M_4, M_5 were calculated, one model per each output class o . Each of these models is the result of training as a binary classifier, i.e. a training to answer “yes” or “no” to the question “does this data item belong to the output class o ?”. To classify a given data item, i.e. to answer the question “to which class does this data-item belong?” we followed different methods yielding comparable results.

4 Determining the class to which a test example belongs

To check whether a test example T belongs to a given class O_j , we check whether the relation $O_j \leq T$ holds. This relation holds in model M_j when all the atoms in constant O_j are a subset of the atoms of term T .

We say an example T has m misses if there are m atoms missing from the set of atoms in O_j . Formally, this number of misses is the the discriminant of the relation $O_j \leq T$ in the output model M_j (see [1] or [13] for definitions),

$$m(T, j) = |disc_{M_j}(O_j, T)|$$

where O_j is the constant associated to output class j .

For an ideal training $m = 0$ for a positive example and $m \geq 1$ for a negative example. However, it is often the case that the amount of data available is not sufficient to perfectly train the algebra. In this case we can find the value of misses Λ_j that separates positives from negatives. We have devised two methods that use a validation dataset and one additional method that does not require validation.

4.1 Using a validation dataset

The first method consists on calculating, for each output class j , the average of the false positive and false negative ratios $(FPR_j(m) + FNR_o(j))/2$ as a function of the number of misses m in the validation dataset. The misses cutoff value Λ_j for the output class j is the value of m that minimizes $FPR_j(m) + FNR_j(m)$ (see 7 (e) and (f) for class 2 as an example),

$$\Lambda_j = \operatorname{argmin}_m (FPR_j(m) + FNR_j(m)) \quad (14)$$

The assigned output class can then be calculated with the help of the cutoff Λ_j as

$$\operatorname{output}(T) = \operatorname{argmin}_j \left(\frac{m(T, j) - \Lambda_j}{|M_j|} \right), \quad (15)$$

where $|M_j|$ is the number of atoms in the model.

The second method computes, for each output class j , the probability to belong to the positive class given the observed number of misses $m(T, j)$, $p_j(+|m)$. This distribution $p_j(+|m)$ can be calculated from the measured distribution of misses in the positive class $p_j(m|+)$ (Figure 7a,b) and the measured distribution of misses $p_j(m|-)$ in the negative classes using the validation dataset (Figure 7c,d) as

$$p_j(+|m) = \frac{p_j(m|+)p_j(+)}{p_j(m|+)p_j(+) + p_j(m|-)p_j(-)}. \quad (16)$$

Using this conditional probability we can compute which is the output class of the example T as

$$\operatorname{output}(T) = \operatorname{argmax}_j p_j(+|m(T, j)), \quad (17)$$

depicted in Figure 7g,h. where $p_o(+)$ and $p_o(-)$ were considered both equal to $1/2$, reflecting the fact that we are going to have no information on the actual distribution of classes in a test set. Other choices would also be justifiable, such as $p_o(+)=1/5$ and $p_o(-)=4/5$ that assume items are distributed with equal probability on each output class.

An advantage of the second method is that a confidence level for the output class assignment can be calculated in a quite straightforward manner.

4.2 Without validation

The third method does not require a validation dataset. The method consists of comparing the cumulative distributions of misses of different output classes in the test set itself:

$$q(T, j, E()) = \sum_{m=0}^{m(T,j)} \frac{E(m)}{E}, \quad (18)$$

where $E(m)$ is the number of test set items that produce m misses and E is the size of the test set, i.e. the number of test set items. The output class is then selected with:

$$\text{output}(T) = \text{argmin}_j q(T, j, E()). \quad (19)$$

This method does not need validation data but it assumes we have a sufficiently large test set at the time of predicting each label. In activity recognition we may be interested in real time predictions, so we may need a method that works with no test set or an small test set. To address this we can adapt this method to classify points individually by using the cumulative distribution of a validation set. This means simply calculating $q(T, j, E())$ in the **validation set** and then classifying points in the test set by computing for each one only its number of misses and assigning to it a label using $\text{argmin}_j q(T, j, E())$, that is, by selecting the class that has the best score according to the validation set's statistics.

It is clear from the size of the cutoffs, that are quite far from 1, that the training data was insufficient to perfectly train any of the output classes. This, however, does not mean that sufficient separation was not achieved. In fact, from output classes such as class number 5, perfect separation was achieved. Also it is worth noticing that the cutoff value was large compared to 1 but it was small in relative terms, i.e. when comparing the minimum number of misses needed with the total number of atoms in the models.

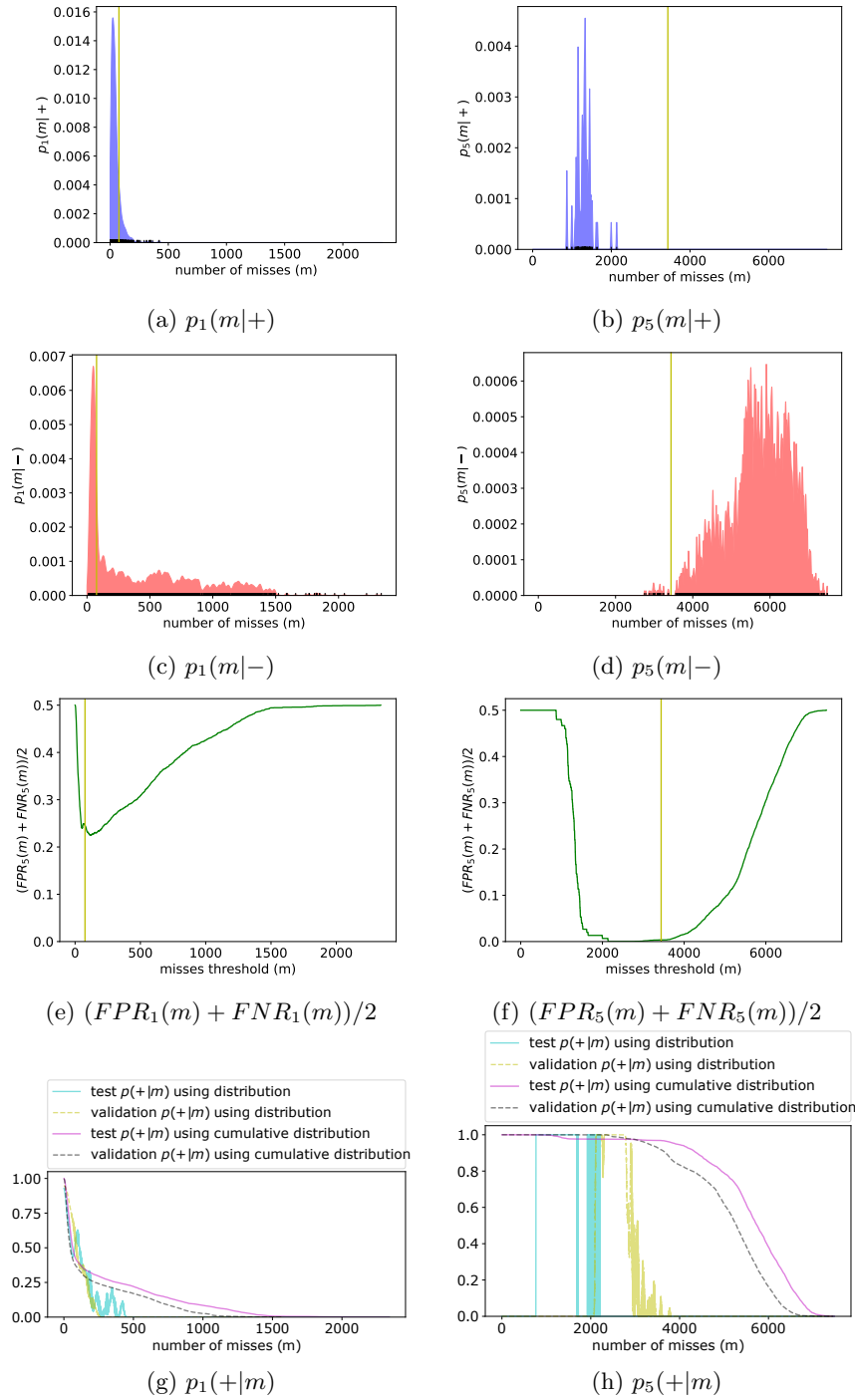


Figure 7: Distributions as function of the number of misses. In blue for the true positive class and in red for the true negative class (in test set). The left column refers to class 1 (Stand), for which AML obtains a moderate separation, and the right one to class 5 (Lie) where AML achieves a perfect separation. In green the average error as a function of a cutoff = m . The number of misses is between 0 and the maximum number of atoms of the respective models M_1 and M_5 . The yellow line is the cutoff selected using the validation set. The last row shows $p(+|m)$ with our different methods in the validation and test sets.

5 Classification Results

To compare the results with AML we used a Decision tree (DT) Classifier and a Multilayer Perceptron (MLP) with one hidden layer containing 64 neurons. For training we used a batch size of 32 with Adam as optimizer, a learning rate of 0.0001, and a weight decay of 0.0005. Further we decayed the learning rate every 5 epochs by a factor of 0.95 and used class weights inversely proportional to class frequency. We trained and tested this model once with the the continuous (not clustered) feature values and once with clustered feature values. The Confusion Matrix for both approaches are given in Figure 10, Figure 11, and Figure 8b.

5.1 Metrics

Since, we are working on a multi class dataset we can compute Precision, Recall and F_1 -Score for each class. Taking one of the confusion matrices C of Figure 8, we define True Positive (TP), False Positive (FP), False Negative (FN), True Negative (TN) for a class i as in [14]:

$$\begin{aligned}
 TP_i &:= C_{i,i} & FN_i &:= \sum_{j=1, j \neq i}^n C_{j,i} \\
 FP_i &:= \sum_{j=1, j \neq i}^n C_{i,j} & TN_i &:= \sum_{j=1, j \neq i}^n \sum_{k=1, k \neq i}^n C_{j,k}
 \end{aligned} \tag{20}$$

where n is the number of classes. Different to [14], we define Precision, Recall and F_1 -Score for each class i and the corresponding averaged metrics as:

$$\begin{aligned}
 P_i &:= \frac{TP_i}{TP_i + FP_i} & P &:= \frac{1}{n} \sum_{i=1}^n P_i \\
 R_i &:= \frac{TP_i}{TP_i + FN_i} & R &:= \frac{1}{n} \sum_{i=1}^n R_i \\
 F_{1i} &:= 2 \frac{P_i \cdot R_i}{P_i + R_i} & F_1 &:= \frac{1}{n} \sum_{i=1}^n F_{1i}
 \end{aligned} \tag{21}$$

The so defined F_1 score is also known as Macro F_1 [15]. Since Macro F_1 averages over all classes, all classes get penalized equally. Therefore, the score drops significantly if the classifier does not work well for one class but is good for all other classes (see Figure 11).

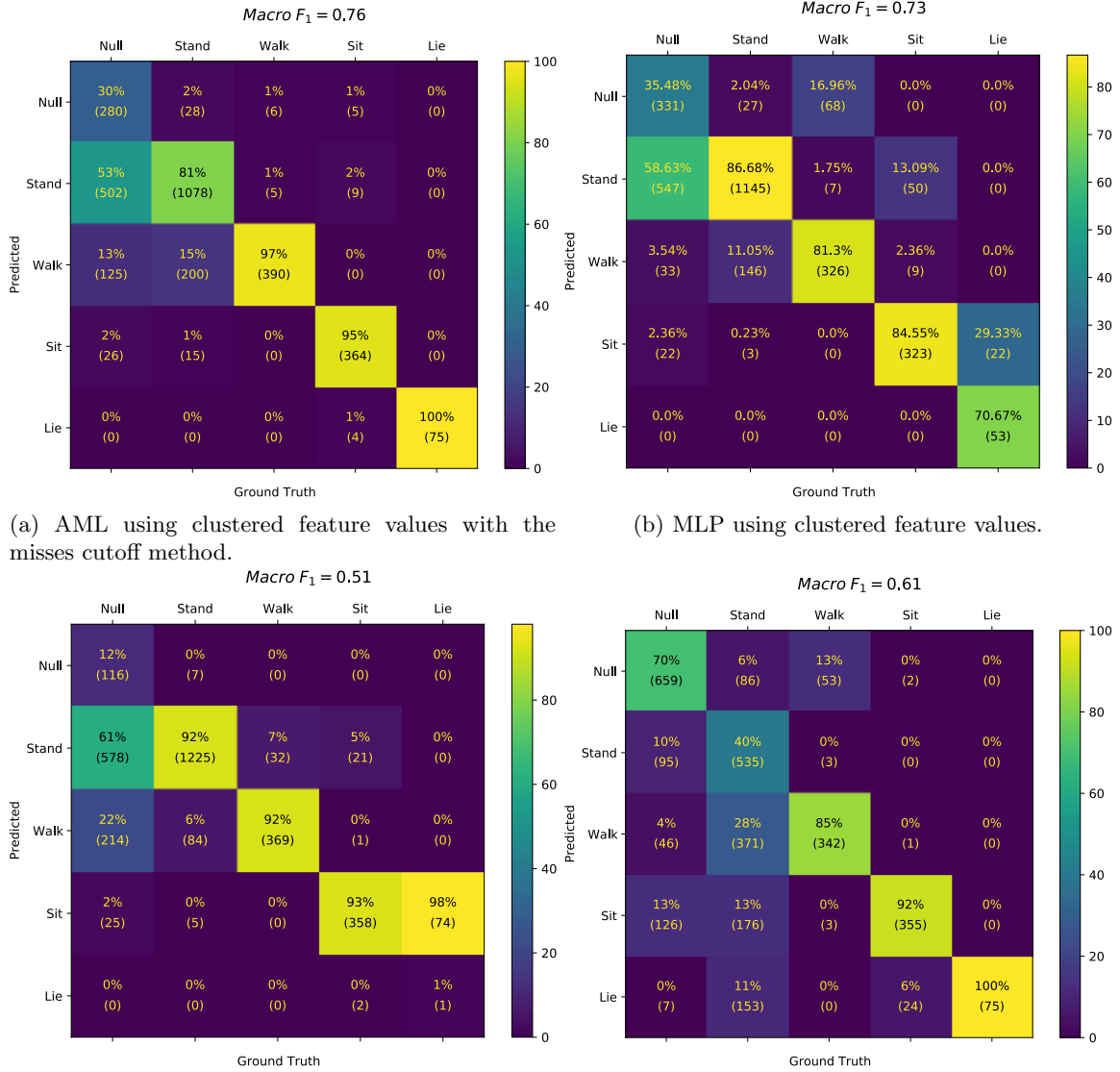
5.2 Overall Results

In Table 2 shows the average Precision, Recall and F_1 -Score for each approach described before.

Method	avg. Precision	avg. Recall	avg. F_1 -Score
DT with clustered feature values	0.51	0.57	0.52
MLP with continuous feature values	0.72	0.78	0.65
MLP with clustered feature values	0.72	0.79	0.73
AML with misses cutoff	0.81	0.79	0.76
AML with non-cumulative misses distribution	0.58	0.65	0.51
AML with cumulative misses distribution	0.78	0.59	0.61

Table 2: Results for the various methods we used including three based on AML.

As we can see by the performance of the DT Classifier in Table 2 this task is not that easy. In addition to the clustered values which makes the task even harder, previous works as [8, 9, 16] mostly used a leave-session-out approach instead of our leave-person-out approach described in section 1.2. The leave-session-out approach used in [8, 9, 16] used for training ADL 1, ADL 2, ADL 3 and Drill of all participants and ADL 4 and ADL 5 of all participants as test set. Our approach is more realistic but also harder due to possible overfitting because the models have all data for only two persons, and therefore less variety. However, this is exactly what was supposed to be the advantage of AML. Probably the best comparison for our MLPs is the work of Li et al. [9]. They used the leave-sessions-out approach, mentioned above and also used a MLP. Whose MLP contains three hidden layers - 2000 neurons each - and got an average F_1 -Score of 66.37%, see Table 3 and 7 in [9]. However, the state of the art for of the OPPORTUNITY challenge is the work from Ordóñez et al. [16]. They also used the above mentioned leave-sessions-out approach, but with Deep-Learning models, i.e. a CNN, and a DeepConvLSTM. The DeepConvLSTM performed best, with a F_1 -Score of 89.5%, see Table 4 in [16].



(c) AML using clustered feature values with $p(+|m)$ misses distribution method. (d) AML using clustered feature values with cumulative misses distribution $q(T, j, E())$ method. This version calculates the cumulative distributions on the unlabeled validation set and then applies them to the test set.

Figure 8: Confusion Matrices to the results given in Table 2

5.3 Results without using a validation set

Since our method that relies on the cumulative distribution can be applied to the test set without using a validation one, we also wanted to explore those results. Thus we explored training on both training and validation set together and then predicting the labels for the test set based on its cumulative misses distribution (which does not need the test's ground truth labels, as explained in Section 4.2). The results

obtained are displayed in Figure 9. As we can see there, the results are very promising. Taking also into account the performance of this cumulative distribution approach when using a validation set (Figure 8d), we think semi-supervised learning is an area where AML can be further explored. This is relevant for Activity Recognition as, similar to other fields, labeling data is cumbersome when compared to obtaining unlabeled data (which with the cumulative method AML can leverage).

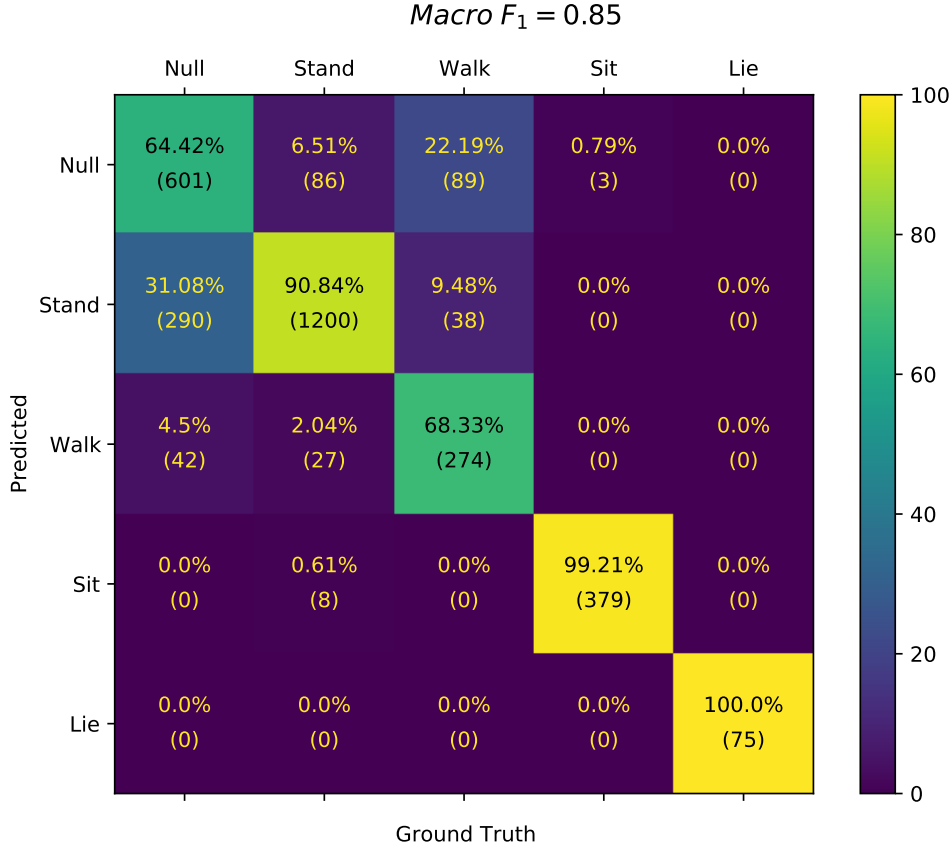


Figure 9: AML using the cumulative misses distribution $q(T, j, E())$ directly on the test set, being trained with both the training and validation set.

6 Conclusion

We have investigated initial methods for applying AML to learning the recognition of human activities from multimodal sensor data time series. As described in the introduction the aim was to (1) develop an initial understanding of how to embed the problem within the algebraic formalism and (2) asses the potential of the method in terms of classification performance.

We have shown that it is possible to use AML in non-discrete problems. Activity recognition data consist of multiple (hundreds) of sensor readings which correspond to continuous intensity quantities. This is the first time this kind of data is used with AML. Furthermore, we were able to do so by embedding in the

algebra each quantity as two discrete linear orders. We found that AML was able to learn from examples on top of the information that we provided as prior formal knowledge; in this case the ordered nature of intensity measurements.

We observed that AML produced atomizations that at some point in the training became larger than the training set itself. Despite that AML generalized without any discernible sign of overfitting.

We also compared AML with standard machine learning approaches: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Nearest Centroid Classifier (NCC), 1-Nearest Neighbours (1 NN), 3-Nearest Neighbours (3 NN) or the MLP with three hidden-layers. Despite this been our first attempt to make AML work in an intrinsically non-discrete problem, results with AML were better than with these methods. Given that this is just our first AML-based approach, we see the results as encouraging as, presumably, there is a lot of room for improvement for AML.

Compared with the state of the art, the results in this study are around 10% behind of the best so far reported using a Deep Learning model designed for the task on the the OPPORTUNITY data set [16]. The difference in performance may arise from our feature extraction preprocess, the discretization or from the different cross validation splits or sensors used. We will investigate this in a next deliverable.

Our study in this first deliverable has the following limitations. First, we are losing information in our discretization method, with our resulting number of features lower than in Li et al. [9]. Second, we reduced the number of used data by only taking a window if more than 75% of the labels are equal and if there is no NaN in a window. This reduced the amount of data we could use to train the models. Third, we have used only one of the possible embeddings with one way to formalize our knowledge of the problem. Finally, with the most basic modes of locomotion (walk, sit stand lay down) problem we addressed what one may consider a toy problem. However, we did include a complex null class which is a type of evaluation is widely reported when using the OPPORTUNITY data. As already explained the focus on this narrow problem will allow us to more easily include physical constraints in the next experiments and to incrementally scale the complexity of the problem.

For the next deliverable we plan to lift some of these limitations. We will use new versions of the AML engine to be able to incorporate more features. We will also study new embedding methods to formalize our formal knowledge of the problem in different ways. These extensions will be tested in a larger number of data sets of human activity, including the full OPPORTUNITY problem with explicit consideration of most included classes. We also plan to compare in more depth with state of the art ML techniques such as deep learning, transformers, etc.

In parallel with the study of how to improve accuracy with different pre-processing and embedding methods, we will make use of the advantages of AML. Specifically, we will 1) Study the case of training and test with very different stats, 2) Study different degrees of unbalance in the data. Additional, we will compare / extend the approach to different datasets, and analyze how fast it can be deployed respect to DL. In addition to accuracy results and deployment times, we will study the atoms of the model to gain insight in what is being learned in the AML approach.

7 Appendix

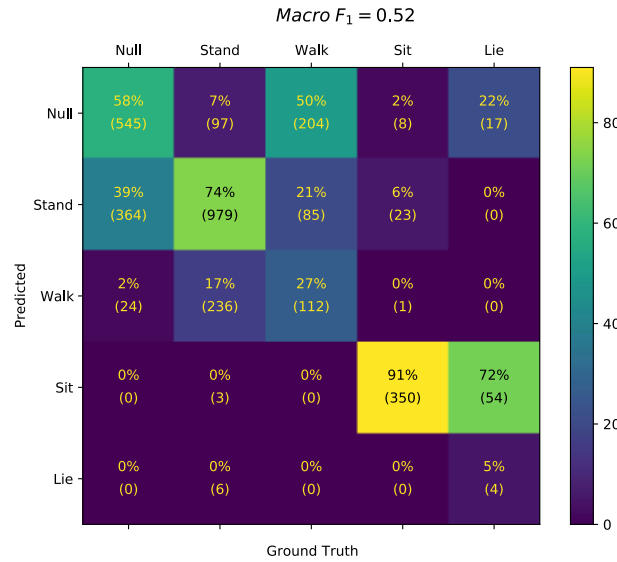


Figure 10: Decision tree using clustered feature values.

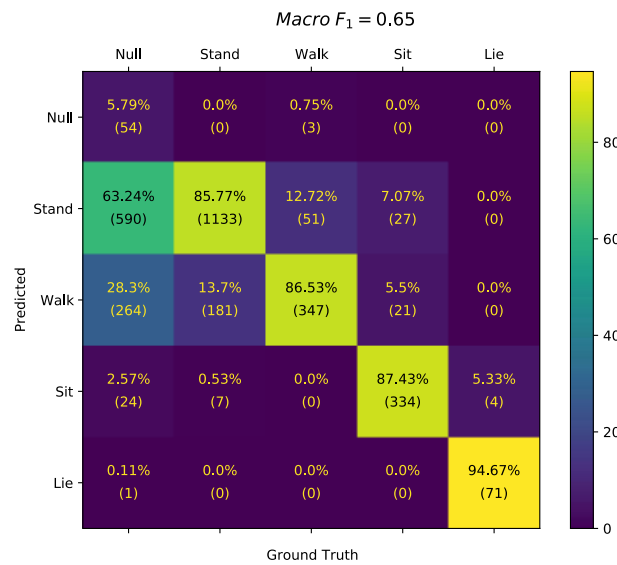


Figure 11: MLP using continuous feature values.

```

1 /* ALGEBRAIC AI 2021 */
2 void descriptor::loadExamplesFor_MOTION_SENSORS_DFKI(problemDescriptor * problemData) {
3
4     int examples = problemData->examples;
5     descriptor * D = this;
6
7     #define DFKI_SENSORS_VARIABLE_RANGE 16 /* for 0 to 15 */
8     #define DFKI_SENSORS_ARRAY_SIZE 353
9

```

```

10 #define DFKI_SENSORS_ARRAY_OUTPUT 352
11 #define DFKI_SENSORS_NUMSENSORS 352
12 #define DFKI_SENSORS_ARRAY_NUMLINES_TRAIN 13570
13 #define DFKI_SENSORS_ARRAY_NUMLINES_TEST 3112
14 #define DFKI_CLUSTER_ARRAY_FULL_VECTOR_SIZE_TRAIN (DFKI_SENSORS_ARRAY_SIZE *
    DFKI_SENSORS_ARRAY_NUMLINES_TRAIN)
15 #define DFKI_CLUSTER_ARRAY_FULL_VECTOR_SIZE_TEST (DFKI_SENSORS_ARRAY_SIZE *
    DFKI_SENSORS_ARRAY_NUMLINES_TEST)
16 {
17     int _i = 1;
18     int i = 0;
19     int k = 0;
20     desc * aux = null;
21     desc * auxI = null;
22     int row = 0;
23     unsigned int sensorReading = 0;
24     int outputValueTested = problemData->data_auxfield_A;
25     int mod = 15;
26     char LEx[10];
27     char Gx[10];
28
29     arrayAllocator * theArrayAllocator = &D->allocator->arrays;
30     fileWrapper fw = DFKI_loadSensorsFile(not testSet, theArrayAllocator);
31     unsigned int * allfile = (unsigned int*) fw.data;
32
33     if (HEADER("Sensors")) {
34         CV("LE0", DFKI_SENSORS_NUMSENSORS);
35         CV("LE1", DFKI_SENSORS_NUMSENSORS);
36         CV("LE2", DFKI_SENSORS_NUMSENSORS);
37         CV("LE3", DFKI_SENSORS_NUMSENSORS);
38         CV("LE4", DFKI_SENSORS_NUMSENSORS);
39         CV("LE5", DFKI_SENSORS_NUMSENSORS);
40         CV("LE6", DFKI_SENSORS_NUMSENSORS);
41         CV("LE7", DFKI_SENSORS_NUMSENSORS);
42         CV("LE8", DFKI_SENSORS_NUMSENSORS);
43         CV("LE9", DFKI_SENSORS_NUMSENSORS);
44         CV("LE10", DFKI_SENSORS_NUMSENSORS);
45         CV("LE11", DFKI_SENSORS_NUMSENSORS);
46         CV("LE12", DFKI_SENSORS_NUMSENSORS);
47         CV("LE13", DFKI_SENSORS_NUMSENSORS);
48         CV("LE14", DFKI_SENSORS_NUMSENSORS);
49
50         CV("G0", DFKI_SENSORS_NUMSENSORS);
51         CV("G1", DFKI_SENSORS_NUMSENSORS);
52         CV("G2", DFKI_SENSORS_NUMSENSORS);
53         CV("G3", DFKI_SENSORS_NUMSENSORS);
54         CV("G4", DFKI_SENSORS_NUMSENSORS);
55         CV("G5", DFKI_SENSORS_NUMSENSORS);
56         CV("G6", DFKI_SENSORS_NUMSENSORS);
57         CV("G7", DFKI_SENSORS_NUMSENSORS);
58         CV("G8", DFKI_SENSORS_NUMSENSORS);
59         CV("G9", DFKI_SENSORS_NUMSENSORS);
60         CV("G10", DFKI_SENSORS_NUMSENSORS);
61         CV("G11", DFKI_SENSORS_NUMSENSORS);
62         CV("G12", DFKI_SENSORS_NUMSENSORS);
63         CV("G13", DFKI_SENSORS_NUMSENSORS);
64         CV("G14", DFKI_SENSORS_NUMSENSORS);
65     }
66
67     if (HEADER("Outputs")) {
68         CV("0", 6);
69     }
70
71     if (HEADER("Inside feature separation")) {

```

```

72     REGION = 0;
73     STORE = true;
74
75     EXC(T("LE0", _i), T("LE1", _i));
76     EXC(T("LE1", _i), T("LE2", _i));
77     EXC(T("LE2", _i), T("LE3", _i));
78     EXC(T("LE3", _i), T("LE4", _i));
79     EXC(T("LE4", _i), T("LE5", _i));
80     EXC(T("LE5", _i), T("LE6", _i));
81     EXC(T("LE6", _i), T("LE7", _i));
82     EXC(T("LE7", _i), T("LE8", _i));
83     EXC(T("LE8", _i), T("LE9", _i));
84     EXC(T("LE9", _i), T("LE10", _i));
85     EXC(T("LE10", _i), T("LE11", _i));
86     EXC(T("LE11", _i), T("LE12", _i));
87     EXC(T("LE12", _i), T("LE13", _i));
88     EXC(T("LE13", _i), T("LE14", _i));
89
90     EXC(T("G1", _i), T("G0", _i));
91     EXC(T("G2", _i), T("G1", _i));
92     EXC(T("G3", _i), T("G2", _i));
93     EXC(T("G4", _i), T("G3", _i));
94     EXC(T("G5", _i), T("G4", _i));
95     EXC(T("G6", _i), T("G5", _i));
96     EXC(T("G7", _i), T("G6", _i));
97     EXC(T("G8", _i), T("G7", _i));
98     EXC(T("G9", _i), T("G8", _i));
99     EXC(T("G10", _i), T("G9", _i));
100    EXC(T("G11", _i), T("G10", _i));
101    EXC(T("G12", _i), T("G11", _i));
102    EXC(T("G13", _i), T("G12", _i));
103    EXC(T("G14", _i), T("G13", _i));
104
105    STORE = false;
106 }
107
108
109 if (HEADER("Feature separation")) {
110     REGION = 0;
111     STORE = true;
112
113     for (k = 0; k < DFKI_SENSORS_NUMSENSORS; k++) {
114         aux = R("LE0", k);
115         aux = M(aux, R("LE1", k));
116         aux = M(aux, R("LE2", k));
117         aux = M(aux, R("LE3", k));
118         aux = M(aux, R("LE4", k));
119         aux = M(aux, R("LE5", k));
120         aux = M(aux, R("LE6", k));
121         aux = M(aux, R("LE7", k));
122         aux = M(aux, R("LE8", k));
123         aux = M(aux, R("LE9", k));
124         aux = M(aux, R("LE10", k));
125         aux = M(aux, R("LE11", k));
126         aux = M(aux, R("LE12", k));
127         aux = M(aux, R("LE13", k));
128         aux = M(aux, R("LE14", k));
129
130         aux = M(aux, R("G0", k));
131         aux = M(aux, R("G1", k));
132         aux = M(aux, R("G2", k));
133         aux = M(aux, R("G3", k));
134         aux = M(aux, R("G4", k));
135         aux = M(aux, R("G5", k));

```

```

136     aux = M(aux, R("G6", k));
137     aux = M(aux, R("G7", k));
138     aux = M(aux, R("G8", k));
139     aux = M(aux, R("G9", k));
140     aux = M(aux, R("G10", k));
141     aux = M(aux, R("G11", k));
142     aux = M(aux, R("G12", k));
143     aux = M(aux, R("G13", k));
144     aux = M(aux, R("G14", k));
145
146     *D ^ aux;
147
148     EXC(F("LE0", k), aux);
149     EXC(F("LE1", k), aux);
150     EXC(F("LE2", k), aux);
151     EXC(F("LE3", k), aux);
152     EXC(F("LE4", k), aux);
153     EXC(F("LE5", k), aux);
154     EXC(F("LE6", k), aux);
155     EXC(F("LE7", k), aux);
156     EXC(F("LE8", k), aux);
157     EXC(F("LE9", k), aux);
158     EXC(F("LE10", k), aux);
159     EXC(F("LE11", k), aux);
160     EXC(F("LE12", k), aux);
161     EXC(F("LE13", k), aux);
162     EXC(F("LE14", k), aux);
163
164     EXC(F("G0", k), aux);
165     EXC(F("G1", k), aux);
166     EXC(F("G2", k), aux);
167     EXC(F("G3", k), aux);
168     EXC(F("G4", k), aux);
169     EXC(F("G5", k), aux);
170     EXC(F("G6", k), aux);
171     EXC(F("G7", k), aux);
172     EXC(F("G8", k), aux);
173     EXC(F("G9", k), aux);
174     EXC(F("G10", k), aux);
175     EXC(F("G11", k), aux);
176     EXC(F("G12", k), aux);
177     EXC(F("G13", k), aux);
178     EXC(F("G14", k), aux);
179 }
180 STORE = false;
181 }
182
183 {
184     REGION = 0;
185     INC(T("LE1", _i), T("LE0", _i));
186     INC(T("LE2", _i), T("LE1", _i));
187     INC(T("LE3", _i), T("LE2", _i));
188     INC(T("LE4", _i), T("LE3", _i));
189     INC(T("LE5", _i), T("LE4", _i));
190     INC(T("LE6", _i), T("LE5", _i));
191     INC(T("LE7", _i), T("LE6", _i));
192     INC(T("LE8", _i), T("LE7", _i));
193     INC(T("LE9", _i), T("LE8", _i));
194     INC(T("LE10", _i), T("LE9", _i));
195     INC(T("LE11", _i), T("LE10", _i));
196     INC(T("LE12", _i), T("LE11", _i));
197     INC(T("LE13", _i), T("LE12", _i));
198     INC(T("LE14", _i), T("LE13", _i));
199

```

```

200     INC(T("G0", _i), T("G1", _i));
201     INC(T("G1", _i), T("G2", _i));
202     INC(T("G2", _i), T("G3", _i));
203     INC(T("G3", _i), T("G4", _i));
204     INC(T("G4", _i), T("G5", _i));
205     INC(T("G5", _i), T("G6", _i));
206     INC(T("G6", _i), T("G7", _i));
207     INC(T("G7", _i), T("G8", _i));
208     INC(T("G8", _i), T("G9", _i));
209     INC(T("G9", _i), T("G10", _i));
210     INC(T("G10", _i), T("G11", _i));
211     INC(T("G11", _i), T("G12", _i));
212     INC(T("G12", _i), T("G13", _i));
213     INC(T("G13", _i), T("G14", _i));
214 }
215
216
217 for (k = 0; k < examples; ++k) {
218     aux = null;
219     int row = 0;
220
221     row = rand() % DFKI_SENSORS_ARRAY_NUMLINES_TRAIN;
222
223     int label = allfile[row*DFKI_SENSORS_ARRAY_SIZE + DFKI_SENSORS_ARRAY_OUTPUT];
224
225     for (i = 0; i < DFKI_SENSORS_NUMSENSORS; i++) {
226         sensorReading = allfile[row*DFKI_SENSORS_ARRAY_SIZE + i];
227
228         switch(sensorReading) {
229             case 0:
230                 M(aux, F("LEO", i));
231                 break;
232
233             case 15:
234                 M(aux, F("G14", i));
235                 break;
236
237             default:
238                 sprintf(LEx, "LE%d\0", sensorReading);
239                 sprintf(Gx, "G%d\0", sensorReading - 1);
240
241                 M(aux, F(LEx, i));
242                 M(aux, F(Gx, i));
243             }
244         }
245
246     EXTERNAL_REFERENCE = row;
247
248     *D ^ aux;
249
250     if (outputValueTested >= 0) {
251         REGION = 1;
252         if (label == outputValueTested) {
253             INC(F("0", outputValueTested), aux);
254         } else {
255             EXC(F("0", outputValueTested), aux);
256         }
257     } else {
258         REGION = 1;
259         if (label == -outputValueTested) {
260             INC(F("0", -outputValueTested), aux);
261         } else {
262             EXC(F("0", -outputValueTested), aux);
263         }
264     }

```

```
264         REGION = 2;
265         INC(F("0", label), aux);
266         EXC(T(R("0", label)), aux);
267     }
268 }
269
270 }
271
272     DFKI_unloadSensorsFileAndClose(theArrayAllocator, &fw);
273 }
274     TARGET_REGION = 1;
275 }
```

References

- [1] Fernando Martin-Maroto and Gonzalo G de Polavieja. “Algebraic Machine Learning”. In: *arXiv preprint arXiv:1803.05252* (2018).
- [2] Oscar D Lara and Miguel A Labrador. “A survey on human activity recognition using wearable sensors”. In: *IEEE communications surveys & tutorials* 15.3 (2012), pp. 1192–1209.
- [3] Benfeng Wang et al. “Deep-learning-based seismic data interpolation: A preliminary result”. In: *Geophysics* 84.1 (2019), pp. V11–V20.
- [4] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.
- [5] Florian Knoll et al. “Advancing machine learning for MR image reconstruction with an open competition: Overview of the 2019 fastMRI challenge”. In: *Magnetic resonance in medicine* 84.6 (2020), pp. 3054–3070.
- [6] Daniel Roggen et al. “Collecting complex activity datasets in highly rich networked sensor environments”. In: *2010 Seventh international conference on networked sensing systems (INSS)*. IEEE. 2010, pp. 233–240.
- [7] Hesam Sagha et al. “Benchmarking classification techniques using the Opportunity human activity dataset”. In: *2011 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE. 2011, pp. 36–40.
- [8] Ricardo Chavarriaga et al. “The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition”. In: *Pattern Recognition Letters* 34.15 (2013), pp. 2033–2042.
- [9] Frédéric Li et al. “Comparison of feature learning methods for human activity recognition using wearable sensors”. In: *Sensors* 18.2 (2018), p. 679.
- [10] Fernando Moya Rueda and Gernot A Fink. “Learning attribute representation for human activity recognition”. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. 2018, pp. 523–528.
- [11] Bo Zhou and Paul Lukowicz. “Snacap: snacking behavior monitoring with smart fabric mechanomyography on the temporalis”. In: *Proceedings of the 2020 International Symposium on Wearable Computers*. 2020, pp. 96–100.
- [12] Gonzalo G. de Polavieja Fernando Martin-Maroto Nabil Abderrahaman-Elena. *AMLDDL specification v0.1*. 2021.
- [13] Fernando Martin-Maroto and Gonzalo G de Polavieja. “Finite ATomized Semilattices”. In: *arXiv preprint arXiv:102.08050* (2021).
- [14] C Manliguez. “Generalized confusion matrix for multiple classes”. In: URL https://www.researchgate.net/publication/310799885_Generalized_Confusion_Matrix_for_Multiple_Classes, DOI: 10.13140/RG.2.2.31150 (2016).
- [15] Juri Opitz and Sebastian Burst. “Macro f1 and macro f1”. In: *arXiv preprint arXiv:1911.03347* (2019).
- [16] Francisco Javier Ordóñez and Daniel Roggen. “Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition”. In: *Sensors* 16.1 (2016), p. 115.