A novel confidence-based multiclass boosting algorithm for mobile physical activity monitoring

3 authors, including:

Gustaf Hendeby
Linköping University
134 PUBLICATIONS **1,599** CITATIONS

SEE PROFILE

Didier Stricker
Technische Universität Kaiserslautern
489 PUBLICATIONS **8,015** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project — BIONIC - Personalized Body Sensor Networks with Built-In Intelligence for Real-Time Risk Assessment and Coaching of Ageing workers, in all types of working and living environments View project

Project — Gesture Recognition View project

A Novel Confidence-based Multiclass Boosting Algorithm for Mobile Physical Activity Monitoring

Attila Reiss · Gustaf Hendeby · Didier Stricker

Abstract This paper addresses one of the main challenges in physical activity monitoring, as indicated by recent benchmark results: The difficulty of the complex classification problems exceeds the potential of existing classifiers. Therefore, this paper proposes the ConfAdaBoost.M1 algorithm. This algorithm is a variant of the AdaBoost.M1 that incorporates well established ideas for confidence-based boosting. ConfAdaBoost.M1 is compared to the most commonly used boosting methods using benchmark datasets from the UCI machine learning repository. Moreover, it is evaluated on an activity recognition and an intensity estimation problem, including a large number of physical activities from the recently released PAMAP2 dataset. The presented results indicate that the proposed ConfAdaBoost.M1 algorithm significantly improves the classification performance on most of the evaluated datasets, especially for larger and more complex classification tasks. Finally, two empirical studies are designed and carried out to investigate the feasibility of ConfAdaBoost.M1 for physical activity monitoring applications in mobile systems.

A. Reiss
Chair of Sensor Technology
University of Passau
E-mail: attila.reiss@uni-passau.de

G. Hendeby
Department of Electrical Engineering
Linköping University
E-mail: hendeby@isy.liu.se; and also
Department of Sensor & EW Systems,
Swedish Defence Research Agency (FOI)

D. Stricker
Department of Augmented Vision
German Research Center for Artificial Intelligence (DFKI)
E-mail: didier.stricker@dfki.de

1 Introduction

Wearable computing is an emerging research field. With recent progress in wearable sensing it becomes reasonable for individuals to wear various sensors all day. Physical activity monitoring is an application area strongly benefiting from this progress. The goals of activity monitoring systems are among others to tell the type of physical activity an individual performs, and to estimate the duration and intensity of the activity performed. With the information obtained this way, the individual's daily routine can be described. One of the strong motivations to achieve these goals comes from healthcare: to be able to tell if individuals are performing enough physical activity — with respect to *e.g.* health recommendations as given by the American College of Sports Medicine and the American Heart Association [14] — to maintain or even promote their health.

Recognition of basic physical activities (such as walk, run or cycle) and basic postures (lie, sit, stand) is well researched [6, 18, 19]. It was shown that good recognition performance can be achieved even with just one 3D-accelerometer and simple classifiers. Moreover, the intensity estimation of these basic activities has been the focus of recent studies, *e.g.* in [21, 38]. However, since these methods only consider a limited set of similar activities, they only apply to specific scenarios. Therefore, current research in this area focuses among others on increasing the number of activities to recognize, with examples of *e.g.* everyday, household or sport activities. This can for instance be achieved by introducing new classification techniques.

The use of meta-level classifiers for activity monitoring problems is not as widespread as using different base-level classifiers. However, comparing base-level and meta-level classifiers on different activity recognition tasks shows that meta-level classifiers (such as boosting, bagging, plurality voting, etc.) outperform base-level classifiers [26]. A complex activity recognition problem including 13 different physical activities is used to evaluate the most widely used base-level (decision trees, *k-nearest neighbors* (kNN), *support vector machines* (SVM) and naive Bayes classifiers) and meta-level (bagging, boosting) classifiers [30]. Best performance was achieved with a boosted decision tree classifier. Recently, a new dataset for physical activity monitoring was introduced and made publicly available: the PAMAP2 dataset [28,29]. Different classification problems have been defined and benchmarked on this dataset, confirming that using a boosted C4.5 decision tree classifier is one of the most promising methods.

The boosted decision tree classifier has — apart from good performance results — further benefits: it is a fast classification algorithm with a simple structure, and is therefore easy to implement. These benefits are especially important for activity monitoring applications since they are usually running on mobile, portable systems for everyday usage, thus the available computational power is limited. Previous work showed the feasibility of using boosted decision tree classifiers for activity recognition on a mobile platform [27]. Moreover, boosting decision trees has been widely and successfully used in other research fields, *e.g.* recently in multitask learning [7]. Therefore, considering all the above mentioned benefits, this work focuses on using boosting, and in particular boosted decision tree classifiers for physical activity monitoring.

The benchmark results on the PAMAP2 dataset [28,29] indicate that the difficulty of the more complex tasks exceed the potential of existing classifiers. Moreover, results presented in [31] show rather low performance when fully simulating how the most common classifiers perform in everyday life scenarios of physical activity monitoring. Therefore, there is a reasonable demand for modifying and improving existing algorithms. This paper proposes a confidence-based extension of the well-known AdaBoost.M1 algorithm, called ConfAdaBoost.M1. It builds on established ideas of existing boosting methods, combining some of their benefits. Overall, the paper provides the following contributions:

1. The paper introduces the ConfAdaBoost.M1 algorithm: a confidence-based extension of the well known Ada-Boost.M1 algorithm. ConfAdaBoost.M1 is a direct multiclass classification technique. Moreover, it uses the information about how confident the weak learners are to predict the class of the instances.
2. A thorough evaluation is presented, comparing Conf-AdaBoost.M1 to the most commonly used existing boos-

ting methods. Experiments performed on 8 different datasets from the UCI machine learning repository are presented. Moreover, evaluation on an activity recognition task and an intensity estimation task (both defined on the PAMAP2 dataset) is carried out. Overall, these experiments show that ConfAdaBoost.M1 significantly improves the results of previous boosting algorithms.
3. Empirical studies are described and carried out to investigate the feasibility of the novel ConfAdaBoost.M1 algorithm for mobile physical activity monitoring applications. These studies show the applicability of Conf-AdaBoost.M1 for both online activity monitoring and the personalisation of such systems on mobile devices.

The rest of this paper is organised as follows. Section 2 gives an overview of existing boosting algorithms, highlighting their benefits and drawbacks. The ConfAdaBoost.M1 algorithm is introduced in Section 3. In Section 4 the new algorithm is evaluated on various benchmark datasets from the UCI machine learning repository, comparing it to the most commonly used existing boosting methods. Section 5 presents the evaluation on a complex activity recognition and intensity estimation problem defined on the PAMAP2 dataset. The main motivation for presenting the ConfAda-Boost.M1 algorithm is the better performance it achieves, compared to existing algorithms, on activity monitoring classification tasks. Section 6 describes two empirical studies performed to examine the feasibility of the proposed Conf-AdaBoost.M1 algorithm for mobile physical activity monitoring systems. Finally, the paper is summarised in Section 7.

This paper is an extended and significantly revised version of [32]. The main novelty of this paper is the investigation of the proposed ConfAdaBoost.M1 algorithm's feasibility for mobile physical activity monitoring systems. Two empirical studies are presented here, along with various insights related to this topic. Moreover, Section 2 includes an extended description of existing boosting methods, while Section 5 describes the applied data processing chain for physical activity monitoring in more detail, compared to the original publication [32].

2 Boosting methods: related work, concepts

Boosting is a widely used and very successful technique for solving classification problems. The idea behind boosting is to iteratively learn weak classifiers by manipulating the training dataset, and then combine the weak classifiers into a final strong classifier. Boosting was introduced in the computational learning theory literature in the early and mid 90's [9,10,34]. To improve a single classifier (weak learner), the first versions of boosting trained additional similar classifiers on filtered versions of the training dataset and pro-

---

Algorithm 1 Real AdaBoost

Require: Training dataset of N instances: $(\underline{x}_i, y_i)\ i = 1, \ldots, N$ ($\underline{x}_i$: feature vector, $y_i \in \{-1, +1\}$)
       New instance to classify: $\underline{x}_n$

1: **procedure** TRAINING($(\underline{x}_i, y_i)\ i = 1, \ldots, N$)
2:     Assign equal weight to each training instance: $w_i = \frac{1}{N}, i = 1, \ldots, N$
3:     **for** $t \leftarrow 1, T$ **do**
4:         Fit weak learner on the weighted dataset to obtain a class probability estimate: $p_t(\underline{x}) = \hat{P}_w(y = 1|\underline{x}) \in [0, 1]$
5:         Compute $f_t(\underline{x}) = \frac{1}{2}\log\frac{p_t(\underline{x})}{1 - p_t(\underline{x})}$
6:         **for** $i \leftarrow 1, N$ **do**
7:            $w_i \leftarrow w_i e^{-y_i f_t(\underline{x}_i)}$
8:         **end for**
9:         Normalize the weight of all instances so that $\sum_i w_i = 1$
10:    **end for**
11: **end procedure**

12: **procedure** PREDICTION($\underline{x}_n$)
13:     The output class is: sign$[\sum_{t=1}^{T} f_t(\underline{x}_n)]$
14: **end procedure**

---

duced a majority vote, thus "boosting" the performance ([9, 34]). The adaptive boosting algorithm — called AdaBoost — evolved from these algorithms [10], and became the most commonly used technique of boosting, from which many versions have been developed. Moreover, AdaBoost is considered as one of the most important ensemble methods, and is named one of the top 10 data mining algorithms by [42].

2.1 Binary classification

The fundamental idea of the boosting technique can be outlined the following way [10]: Assume that a training dataset of N instances is given: $(\underline{x}_i, y_i)\ i = 1, \ldots, N$ ($\underline{x}_i$ is the feature vector, $y_i \in \{-1, +1\}$). The algorithm trains weak learners, $f_t(\underline{x})$, on weighted versions of the training dataset, giving higher weight to currently misclassified instances. This is performed a predefined number of iterations, T. The final classifier is a linear combination of the weak learners from each iteration, weighted according to their error rate on the training dataset. The first version of the AdaBoost algorithm only uses the binary output of the weak learners, and is thus called Discrete AdaBoost. The Real AdaBoost algorithm [12] is a generalization of the original algorithm that use real-valued predictions of the weak learners rather than the $\{-1, +1\}$ output, as shown in Algorithm 1. The weak learners then return a class probability estimate, $p_t(\underline{x})$, in each boosting iteration, from which the classification rule $f_t(\underline{x})$ is derived. The sign of $f_t(\underline{x})$ gives the classification prediction, and $|f_t(\underline{x})|$ gives a measure of how confident the weak learner is in the prediction. Experiments on various datasets from the UCI machine learning repository [3] show that this confidence-based version of AdaBoost outperforms the original Discrete AdaBoost algorithm [12]. However, both the Discrete and Real AdaBoost are limited to binary classification problems.

Apart from Discrete and Real AdaBoost, further boosting methods have been developed for the binary classification case the past decade. The Discrete and Real AdaBoost algorithms can be interpreted as sequential estimation procedures for fitting an additive logistic regression model, optimizing an exponential criterion which to second order is equivalent to the binomial log-likelihood criterion [12]. Based on this interpretation, the LogitBoost algorithm was introduced, which optimizes a more standard (the Bernoulli) log-likelihood [12]. Moreover, [12] also presents the Gentle AdaBoost algorithm, a modified version of Real AdaBoost. It uses Newton stepping rather than exact optimization at each boosting iteration. Another variant of Real AdaBoost – that uses a weighted emphasis function – is presented in [13], called Emphasis Boost. Finally, the Modest AdaBoost algorithm is mentioned here [40]. It not only considers the updated weight distribution for training a classification rule in each boosting step, but also considers the inverse weight distribution to decrease a weak learner's contribution if it works "too good" on data that has already been correctly classified with high margin. As a result, although the training error decreases slower than for comparable methods, Modest AdaBoost produces less generalization error.

2.2 Pseudo-multiclass classification

The first extensions of AdaBoost for multiclass classification problems can be regarded as pseudo-multiclass solutions: they reduce the multiclass problem into multiple two class problems [35, 36]. One of the most common solutions using binary boosting methods for multiclass problems is AdaBoost.MH [36]. It converts a C class problem into that of estimating a two class classifier on a training set C times as large, by adding a new "feature" which is defined by the class labels. Thus the original number of N instances is expanded into NC instances. On this new, augmented dataset

---

Algorithm 2 AdaBoost.M1

Require: Training dataset of N instances: $(\underline{x}_i, y_i)$ $i = 1, \ldots, N$ ($\underline{x}_i$: feature vector, $y_i \in [1, \ldots, C]$)
        New instance to classify: $\underline{x}_n$
1: **procedure** TRAINING($(\underline{x}_i, y_i)$ $i = 1, \ldots, N$)
2:    Assign equal weight to each training instance: $w_i = \frac{1}{N}$, $i = 1, \ldots, N$
3:    **for** $t \leftarrow 1, T$ **do**
4:        Fit weak learner on the weighted dataset: $f_t(\underline{x}) \in [1, \ldots, C]$
5:        Compute error e_t of model on weighted dataset: $e_t = \sum_{i:y_i \neq f_t(\underline{x}_i)} w_i$
6:        **if** $e_t = 0$ or $e_t \geq 0.5$ **then**
7:            Delete last $f_t(\underline{x})$ and terminate model generation.
8:        **end if**
9:        Compute $\alpha_t = \log \frac{1-e_t}{e_t}$
10:       **for** $i \leftarrow 1, N$ **do**
11:           **if** $y_i \neq f_t(\underline{x}_i)$ **then**
12:               $w_i \leftarrow w_i e^{\alpha_t}$
13:           **end if**
14:       **end for**
15:       Normalize the weight of all instances so that $\sum_i w_i = 1$
16:    **end for**
17: **end procedure**

18: **procedure** PREDICTION($\underline{x}_n$)
19:    Set zero weight to all classes: $\mu_j = 0$, $j = 1, \ldots, C$
20:    **for** $t \leftarrow 1, T$ **do**
21:        Predict class with current model: $c = f_t(\underline{x}_n)$
22:        $\mu_c \leftarrow \mu_c + \alpha_t$
23:    **end for**
24:    The output class is $\arg \max_j \mu_j$ $j = 1, \ldots, C$
25: **end procedure**

---

a binary AdaBoost method (*e.g.* Discrete or Real AdaBoost) can then be applied.

There exist other solutions to reduce the multiclass problem into multiple binary classification problems. In [35] *error-correcting output codes* (ECOC) were combined with the original binary AdaBoost method to solve multiclass problems, resulting in the AdaBoost.MO algorithm. In [12] it was shown how the binary LogitBoost algorithm can be applied for the multiclass case by introducing a "class feature" similar to the AdaBoost.MH method. In [35] experimental results are given comparing a few pseudo-multiclass algorithms on a set of benchmark problems from the UCI repository, which show that Real AdaBoost.MH (the extension of the binary Real AdaBoost algorithm for the multiclass case using the AdaBoost.MH technique) performs best amongst these methods.

However, reducing the multiclass classification problem into multiple two class problems has several drawbacks. For instance, as the class label becomes a regular feature in the AdaBoost.MH method, its importance is significantly reduced. AdaBoost.MH is an asymmetric strategy, building separate two class models for each individual class against the pooled complement classes. Pooling classes can produce more complex decision boundaries that are difficult to approximate, while separating class pairs could be relatively simple [12]. Moreover, pseudo-multiclass algorithms might create resource problems by increasing (basically multiply-

ing) *e.g.* training time or memory required, especially for problems with a large number of classes. Therefore, to overcome these drawbacks direct multiclass extensions of the AdaBoost method should be developed and investigated. Nevertheless, pseudo-multiclass methods will remain interesting since they can be used for the multiclass multilabeled case: when instances may belong to more than one class [36]. An application scenario for this case is *e.g.* text categorization: one document can be assigned to more than one topic. If the goal is to predict all and only all of the correct labels, the AdaBoost.MH algorithm is a valid solution.

2.3 Multiclass classification

The first direct multiclass extension of the original Ada-Boost algorithm, AdaBoost.M1, was introduced in [10] and is the most widely used multiclass boosting method. It is also the basis of many further variants of multiclass boosting. The AdaBoost.M1 algorithm is shown in Algorithm 2. Similar to the binary AdaBoost methods, it can be used with any weak classifier that has an error rate of less than 0.5. However, this criterion is more restrictive than for binary classification, where an error rate of 0.5 means basically random guessing. In [10] a second multiclass extension of the original AdaBoost algorithm, AdaBoost.M2, was also introduced. In this algorithm the weak classifiers have to minimize a newly introduced pseudo-loss, instead of minimal-

izing the error rate as usually done. The pseudo-loss of the weak classifiers has to be less than 0.5, but this is a much weaker condition than the error rate being less than 0.5. The drawback of AdaBoost.M2 is that classifiers have to be re-designed in order to be used as weak learners within this algorithm, since almost all traditionally used classifiers minimize the error rate and not the new pseudo-loss.

In [5, 46] another way to overcome the restriction on the weak learner's error rate is shown by adding a constant taking the number of classes (C) into account, this way relaxing the requirement of the weak classifiers to an error rate of less than random guessing ($1 - \frac{1}{C}$). In [5] the Ada-Boost.M1W algorithm was introduced based on this idea, and proved with experiments its benefits over AdaBoost.M1. The SAMME (*stagewise additive modeling using a multiclass exponential loss function*) algorithm in [46] is based on the same idea. It has an identical structure to AdaBoost.M1, the only difference is how α_t is computed on line 9: $\alpha_t = \log \frac{1-e_t}{e_t} + \log(C - 1)$. It is shown in [46] that the extra term $\log(C-1)$ is not artificial: Similar to the interpretation of AdaBoost in [12], SAMME is equivalent to fitting a forward stage-wise additive model using a multiclass exponential loss function. Obviously when $C = 2$, SAMME reduces to AdaBoost.M1. However, the extra term $\log(C-1)$ is critical in the multiclass case, since in order for α_t to be positive only the condition $(1 - e_t) > \frac{1}{C}$ is required. Therefore, the error rate of the weak learners only has to be better than random guessing rather than 0.5. In [46] the SAMME algorithm was compared to AdaBoost.MH on various benchmark datasets from the UCI repository. It was shown that SAMME's performance is comparable with that of the AdaBoost.MH method, or even slightly better. The SAMME.R variation [45] of the SAMME algorithm uses the probability estimates from the weak classifiers. However, SAMME.R does not keep the structure of AdaBoost.M1: when updating the weights for the training instances only the respective probability estimates are used, the error e_t of the weak learner on the weighted dataset is not considered. Moreover, the SAMME.R algorithm showed overall slightly worse performance results than SAMME on different datasets [45], thus is discarded from further analysis in this work.

Another multiclass boosting method is introduced in [15]: GAMBLE (*gentle adaptive multiclass Boosting learning*) is the generalised version of the binary Gentle AdaBoost algorithm. However, GAMBLE fits a regression model rather than a classification model at each boosting iteration, thus requires several additional steps in order to be used for classification tasks (which is the actual focus of this work). First the class labels have to be encoded (*e.g.* with response encoding), then the regression model is fitted which is then used to obtain the weak classifier. Overall, the training time and computational cost is significantly increased compared to AdaBoost models using directly classification models.

3 ConfAdaBoost.M1

Various boosting algorithms exist and were presented in the previous section. However, there are still classification problems where the difficulty of the task exceeds the potential of existing methods, *e.g.*, in the field of physical activity monitoring [28, 29]. Moreover, experiments presented in this paper show a high error rate on the PAMAP2 physical activity monitoring dataset with selected, commonly used boosting algorithms. Therefore, there is a need for further development of boosting techniques to improve the performance on such complex classification tasks.

This section introduces a new variant of the boosting algorithm, called ConfAdaBoost.M1. It is a confidence-based extension of the AdaBoost.M1 algorithm based on a combination of concepts and ideas used in the previously described boosting methods. First of all it is a direct multiclass classification technique, thus it overcomes the drawbacks of pseudo-multiclass boosting methods. Moreover, it keeps the structure of AdaBoost.M1, thus when already using AdaBoost.M1 in a classification task it can be easily extended to ConfAdaBoost.M1. Furthermore, the new algorithm uses the information about how confident the weak learners are to predict the class of the instances. This approach has been beneficial in both binary (when developing the Real AdaBoost algorithm from Discrete AdaBoost in [12]) and pseudo-multiclass (the improvement of Discrete AdaBoost.MH to Real AdaBoost.MH in [36]) classification. Therefore, this work takes the next step by applying the idea of a confidence-based version of AdaBoost for the direct multiclass classification case. It is worth mentioning that [25] already proposed to modify the prediction step of the AdaBoost.M1 algorithm to allow the voting weights of the weak learners to vary in response to the confidence with which $\underline{x}_n$ (the new instance to be classified) is classified. However, no confidence-based extension of the training part of the AdaBoost.M1 algorithm has previously been proposed.

The main idea of the ConfAdaBoost.M1 algorithm can be described as follows. In the training part of the algorithm the confidence of the classification estimation is returned for each instance by the weak learner, and is then used to compute the new weight of that instance: the more confident the weak learner is in a correct classification the more the weight will be reduced, and the more confident the weak learner is in a misclassification the more the weight will be increased. Moreover, the confidence values are also used in the prediction part of the algorithm: the more confident the weak learner is in a new instance's prediction the more it counts in the output of the combined classifier, as proposed in [25].

The ConfAdaBoost.M1 algorithm is shown in Algorithm 3. The structure of the original AdaBoost.M1 algorithm is kept (*cf.* Algorithm 2), extending it on multiple lines. First

---

Algorithm 3 ConfAdaBoost.M1

Require: Training dataset of N instances: $(\underline{x}_i, y_i)\; i = 1, \ldots, N$ ($\underline{x}_i$: feature vector, $y_i \in [1, \ldots, C]$)
         New instance to classify: $\underline{x}_n$

 1: **procedure** TRAINING($(\underline{x}_i, y_i)\; i = 1, \ldots, N$)
 2:      Assign equal weight to each training instance: $w_i = \frac{1}{N}, i = 1, \ldots, N$
 3:      **for** $t \leftarrow 1, T$ **do**
 4:          Fit weak learner on the weighted dataset: $f_t(\underline{x}) \in [1, \ldots, C]$
 5:          Compute the confidence of the prediction that instance $\underline{x}_i$ belongs to the predicted class: $p_{ti}, i = 1, \ldots, N$
 6:          Compute error e_t of model on weighted dataset: $e_t = \sum_{i:y_i \neq f_t(\underline{x}_i)} p_{ti} w_i$
 7:          **if** $e_t = 0$ or $e_t \geq 0.5$ **then**
 8:             Delete last $f_t(\underline{x})$ and terminate model generation.
 9:          **end if**
10:          Compute $\alpha_t = \frac{1}{2} \log \frac{1-e_t}{e_t}$
11:          **for** $i \leftarrow 1, N$ **do**
12:             $w_i \leftarrow w_i e^{\left(\frac{1}{2} - \mathbb{I}(y_i = f_t(\underline{x}_i))\right) p_{ti} \alpha_t}$
13:          **end for**
14:          Normalize the weight of all instances so that $\sum_i w_i = 1$
15:      **end for**
16: **end procedure**

17: **procedure** PREDICTION($\underline{x}_n$)
18:      Set zero weight to all classes: $\mu_j = 0, j = 1, \ldots, C$
19:      **for** $t \leftarrow 1, T$ **do**
20:          Predict class with current model: $[c, p_t(\underline{x}_n)] = f_t(\underline{x}_n)$,
            where $p_t(\underline{x}_n)$ is the confidence of the prediction that instance $\underline{x}_n$ belongs to the predicted class c
21:          $\mu_c \leftarrow \mu_c + p_t(\underline{x}_n)\alpha_t$
22:      **end for**
23:      The output class is $\arg\max_j \mu_j \quad j = 1, \ldots, C$
24: **end procedure**

---

of all, after training the weak learner on the weighted dataset (line 4), the confidence of the classification estimation is returned for each instance by this weak learner (line 5). These p_{ti} confidence values are used when computing the error rate of the weak learner (line 6): the more confident the model is in the misclassification the more that instance's weight counts in the overall error rate. The factor $\frac{1}{2}$ on line 10 of the ConfAdaBoost.M1 algorithm is used to compensate the lower e_t compared to the computed error rate of AdaBoost.M1. The p_{ti} confidence values are also used to recompute the weights of the instances. The more confident the weak learner is in an instance's correct classification or misclassification, the more that instance's weight is reduced or increased, respectively (line 12). The factor $\frac{1}{2}$ on line 12 (determined in an empirical study) is applied in addition compared to the original AdaBoost.M1 algorithm, to compensate that weights are modified in both directions before the renormalization of the weights. In the prediction part of ConfAdaBoost.M1 the only modification compared to the AdaBoost.M1 algorithm is that the confidence of the prediction ($p_t(\underline{x}_n)$) is computed (line 20), and then used to adjust the voting weights of the weak learners (line 21).

It should be noted that the $e_t \geq 0.5$ stopping criterion of the original AdaBoost.M1 remains the same in the proposed ConfAdaBoost.M1 algorithm (line 7). This means that, similar to AdaBoost.M1, only classifiers achieving a reasonably high accuracy value can be used as weak learners, thus *e.g.*

decision stumps are not suitable for multiclass problems. However, the stopping criterion of $e_t \geq 0.5$ is less restrictive in ConfAdaBoost.M1, since the computation of the error rate also uses the p_{ti} confidence values, thus the computed e_t is lower. Therefore, when using the same weak learner, ConfAdaBoost.M1 can perform significantly more boosting iterations before stopping compared to AdaBoost.M1, as shown in the experiments of the next sections.

4 Evaluation on UCI datasets

In this section experiments on various datasets from the UCI machine learning repository [3] are presented. These experiments compare the newly introduced ConfAdaBoost.M1 algorithm to the most commonly used existing boosting methods. The first part of this section presents the basic conditions of the experiments, then results are given and discussed.

4.1 Basic Conditions

The experiments were performed on 8 datasets from the UCI repository. The selected benchmark datasets include 3 small datasets: *Glass*, *Iris* [8] and *Vehicle* [37], as well as 5 pre-partitioned larger datasets: *Letter* [11], *Pendigits* [2], *Satim-age*, *Segmentation* and *Thyroid* [23]. The parameters of the

Table 1 Summary of the benchmark datasets used in the experiments. Size of a training and test part listed in case pre-partitioning of the dataset is provided in the UCI repository.

Dataset	Total	#Instances Training	Testing	#Variables	#Classes
Glass	214	—	—	9	6
Iris	150	—	—	4	3
Vehicle	846	—	—	18	4
Letter	20000	16000	4000	16	26
Pendigits	10992	7494	3498	16	10
Satimage	6435	4435	2000	36	6
Segmentation	2310	210	2100	19	7
Thyroid	7200	3772	3428	21	3
PAMAP2_AR	19863	—	—	137	15
PAMAP2_IE	24197	—	—	137	3

used datasets are summarised in Table 1. These datasets were selected with the goal to cover a wide range of scenarios: The size of the datasets ranges from 150 to 20 000 instances, the number of classes ranges from 3 to 26, and the difficulty of the classification problems these datasets define vary a lot as well according to experiments performed on these datasets in previous work (*cf. e.g.* [36,46]). A further selection criterion was to only include datasets which directly provide the features of the classification tasks as attributes, thus no domain knowledge (*e.g.* how to process the provided data, which features should be extracted, *etc.*) of the datasets should be required. Using various datasets from the UCI repository is common practice when introducing a new boosting method and comparing it to existing algorithms. For instance, [46] used 7 different UCI datasets to compare the SAMME algorithm to AdaBoost.MH, and [16] used 23 UCI datasets to compare the proposed AdaBoost.HM algorithm to AdaBoost.M1 and AdaBoost.MH. Finally, using datasets which have been applied before allows a real comparison to previous work.

On the selected datasets, the ConfAdaBoost.M1 algorithm is compared to 4 other existing boosting methods. First of all to AdaBoost.M1 to provide the baseline performance of the experiments (since, as many other boosting variants, ConfAdaBoost.M1 is also an extension of AdaBoost.M1). The proposed confidence-based modification of the prediction step of AdaBoost.M1 in [25] is part of the ConfAdaBoost.M1 algorithm. Therefore, it is of interest to compare to this extension (which will be referred to as QuinlanAdaBoost.M1 hereafter) of the original AdaBoost.M1, to investigate whether possible performance improvements come from only the confidence-based prediction step or the confidence-based extension of both the training and prediction steps, as proposed by ConfAdaBoost.M1. The next boosting method used for comparison is SAMME, since according to [5,46] this direct multiclass extension of AdaBoost.M1 outperforms traditionally used boosting techniques. Finally, the most common pseudo-multiclass classification technique is used for comparison: the Real AdaBoost.MH algorithm. It

performs best amongst the pseudo-multiclass methods and is a confidence-based boosting version similar to ConfAda-Boost.M1.

The C4.5 decision tree classifier [24] is used as weak learner in each of the evaluated boosting methods. This classifier is, together with decision stumps, the most commonly used weak learner for boosting. It also fulfills the requirement of achieving a reasonably high accuracy on the different classification problems (it has an error rate of significantly less than 0.5 on the various datasets, as shown below by the results), thus can be used with the algorithms Ada-Boost.M1, QuinlanAdaBoost.M1 and ConfAdaBoost.M1. Considering confidence-based versions of AdaBoost, the C4.5 decision tree has another benefit: there is no need to modify the C4.5 algorithm, the confidence values of the weak learners' predictions can be directly extracted from the trained decision trees. Assume that a C4.5 decision tree is trained as $f_t(\underline{x})$ weak learner in the ConfAdaBoost.M1 algorithm (Algorithm 3, line 4). The p_{ti} confidence of the prediction that instance $\underline{x}_i$ belongs to the predicted class (Algorithm 3, line 5) can be computed as follows, based on [25]. In the trained C4.5 decision tree a single leaf node classifies $\underline{x}_i$: $c = f_t(\underline{x}_i)$. Let S be the training instances mapped to this leaf, and let S_c be the subset of S belonging to class c. The confidence of the prediction is then:

$$p_{ti} = \left(\sum_{j \in S_c} w_j \right) / \left(\sum_{j \in S} w_j \right). \tag{1}$$

On the 5 larger, pre-partitioned datasets pruned C4.5 decision trees are used. The level of pruning is defined by 5-fold *cross-validation* (CV) on the training part of these datasets, for each of the evaluated boosting methods separately. On the 3 smaller datasets (Glass, Iris and Vehicle), non-pruned C4.5 decision trees are used as weak learners. Between 1 and 500 boosting iterations are evaluated for all algorithms and benchmark datasets (previous work *e.g.* in [5,46] showed that the performance of various boosting algorithms usually levels off at maximum 100 iterations). All results presented below are averages of multiple test runs. On datasets providing a training and test part training is per-
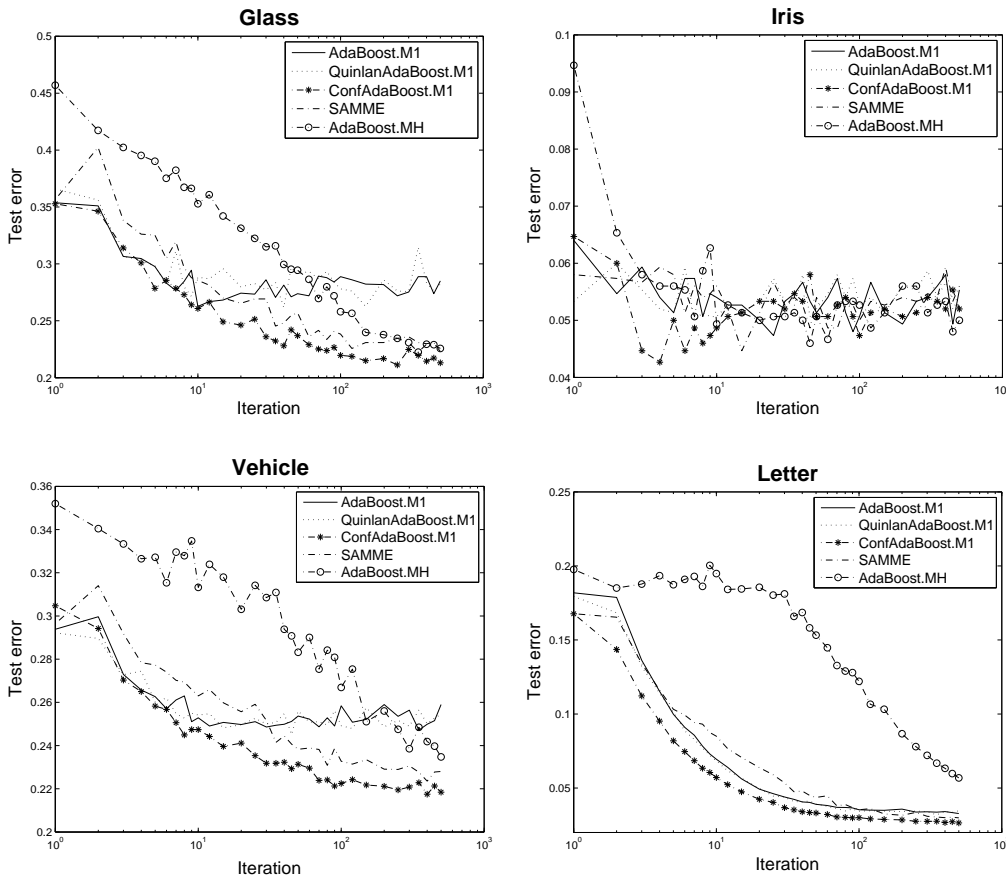
Fig. 1 Test error of the 5 evaluated boosting algorithms on the UCI benchmark datasets Glass, Iris, Vehicle and Letter. The results are averages over 10 test runs.

formed 10 times on the training set, and the trained classifier is then evaluated on the provided test set each time. On datasets without a predefined test part, 10-fold CV is used and performed 10 separate times. All experiments were performed within Matlab, random substreams are used to ensure randomness between different test runs.

4.2 Results and Discussion

The averaged results of the 10 test runs on the selected 8 UCI benchmark datasets are shown in Fig. 1 and 2. The test errors of the 5 evaluated boosting methods are summarised in Table 2. Overall it is clear that the ConfAdaBoost.M1 algorithm performed best in the experiments: on 7 out of 8 datasets there is a noticeable increase in performance compared to existing boosting methods, while on one dataset (Thyroid) ConfAdaBoost.M1 has essentially the same performance as the other algorithms. According to the results of Table 2, the second best boosting algorithm is SAMME, closely followed by AdaBoost.MH, confirming the results of [46]. The original AdaBoost.M1 and its variation Quin-

lanAdaBoost.M1 performed overall clearly worse, the latter algorithm being slightly but not significantly better.

A statistical significance test (the McNemar test [17] is used to pair-wise compare the predictions of the different methods) indicates that the reduction of the test error rate by ConfAdaBoost.M1 compared to SAMME is significant with p-value 0.01 on the datasets Pendigits and Segmentation, significant with p-value 0.05 on the datasets Letter and Satimage, and that on the remaining datasets no statistical significance was observed. In conclusion, the ConfAdaBoost.M1 algorithm has more potential for improvement the larger the dataset and the more complex the classification problem is. This statement is supported by the results on the PAMAP2 classification tasks in the next section. Moreover, similar observation was made in [36] when comparing Discrete and Real AdaBoost.MH: the confidence-based method had better capability for improvement the larger the datasets were. On the Thyroid dataset on the other hand even AdaBoost.M1 reaches an error rate of less than 1% leaving only a few outlier instances misclassified, thus explaining the minimal (not statistically significant) difference between the results of the 5 algorithms. Furthermore, [12] concludes that interpreting results and slight performance differences
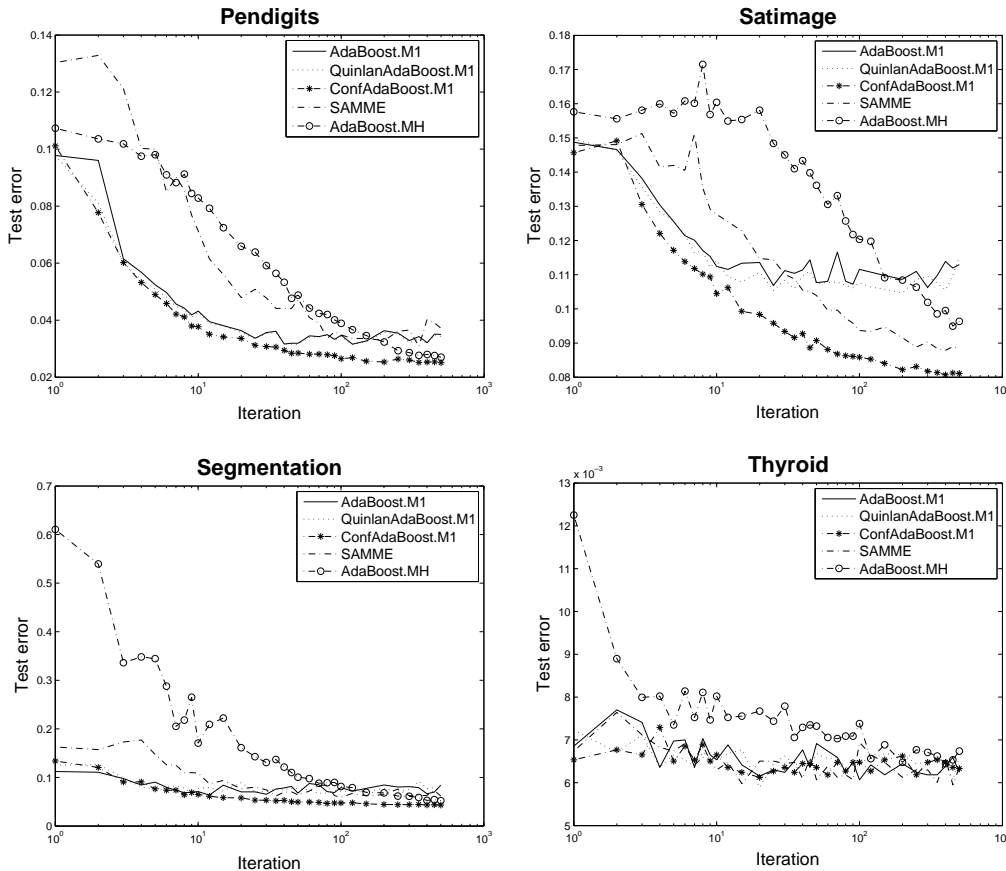
Fig. 2 Test error of the 5 evaluated boosting algorithms on the UCI benchmark datasets Pendigits, Satimage, Segmentation and Thyroid. The results are averages over 10 test runs.

on rather small datasets is difficult since it can occur due to sampling fluctuations, while on the larger datasets clearer trends are observable.

One of the main reasons why AdaBoost.M1 and QuinlanAdaBoost.M1 performs significantly worse than the other methods is that they reach the stopping criterion of $e_t \geq 0.5$ quickly. This can be observed especially on the results of the datasets Glass, Vehicle or Satimage: the test error decreases at the beginning but levels off already at around 10 to 20 boosting iterations, no further improvement can be reached with the increase of the number of boosting rounds. This effect is not observed when using the ConfAdaBoost.M1 algorithm due to the modified computation of the error rate of the weak learners. Another benefit of ConfAdaBoost.M1 over the other methods can be observed *e.g.* on the results of the datasets Vehicle, Letter and Satimage: the test error even at lower numbers of boosting iterations is the lowest when using ConfAdaBoost.M1. This means that for a particular level of accuracy fewer boosting rounds (thus smaller classifier size) are necessary with ConfAdaBoost.M1 compared to existing boosting algorithms. Therefore, the better performance of ConfAdaBoost.M1 can be partially explained by faster convergence (*cf. e.g.* the results of AdaBoost.MH on

the aforementioned datasets). However, this quality is especially beneficial when the available computational resources are limited, which is usually the case for physical activity monitoring applications.

Finally, it is worth to discuss and compare the training time required for creating the different classifiers. Building a decision tree has the time complexity of $\mathcal{O}(DMN \log(N))$, where N is the number of training instances, M is the dimension of the feature vector of a training instance, and D is the average depth of the decision tree [45]. The computational cost of AdaBoost.M1 is then $\mathcal{O}(DMN \log(N)T)$, where T is the number of boosting iterations. The theoretical complexity of the algorithms QuinlanAdaBoost.M1, SAMME and the newly proposed ConfAdaBoost.M1 is similar. The computational cost of AdaBoost.MH is $\mathcal{O}(DMN \cdot \log(N)TC)$, where C refers to the number of classes. During the experiments of this section, the training time of ConfAdaBoost.M1 was comparable to that of SAMME on all 8 evaluated datasets. Compared to these two algorithms, the training time of AdaBoost.M1 and QuinlanAdaBoost.M1 was almost an order of magnitude lower. This can be explained with the early reaching of the stopping criterion, as discussed in the previous paragraph (thus T gets smaller in

Table 2 Comparison of the 5 evaluated boosting algorithms: test error rates [%] on the selected benchmark datasets. The results are averaged over 10 test runs (mean and standard deviation are given), the best performance is shown for each of the methods.

Dataset	AdaBoost.M1	Quinlan-AdaBoost.M1	Conf-AdaBoost.M1	SAMME	AdaBoost.MH
Glass	26.26 ± 1.42	26.17 ± 2.60	**21.12** ± 1.22	22.29 ± 1.38	22.24 ± 1.81
Iris	4.73 ± 0.73	5.00 ± 0.85	**4.27** ± 0.64	4.47 ± 1.22	4.60 ± 0.80
Vehicle	24.72 ± 1.05	24.52 ± 1.10	**21.75** ± 0.44	22.35 ± 1.14	23.48 ± 1.21
Letter	3.28 ± 0.14	3.19 ± 0.15	**2.64** ± 0.11	2.99 ± 0.13	5.68 ± 0.39
Pendigits	3.16 ± 0.27	3.14 ± 0.45	**2.51** ± 0.11	3.08 ± 0.15	2.70 ± 0.08
Satimage	10.63 ± 0.80	10.47 ± 1.01	**8.07** ± 0.15	8.79 ± 0.25	9.50 ± 0.39
Segmentation	6.36 ± 1.03	6.55 ± 1.08	**4.31** ± 0.20	5.92 ± 0.79	5.22 ± 0.78
Thyroid	0.61 ± 0.04	**0.59** ± 0.05	0.61 ± 0.05	0.60 ± 0.06	0.64 ± 0.08
PAMAP2_AR	29.28 ± 1.40	27.90 ± 1.06	**22.22** ± 0.77	27.98 ± 1.34	—
PAMAP2_IE	7.98 ± 1.04	7.73 ± 0.66	**5.60** ± 0.31	7.81 ± 0.60	—

the expression of $\mathcal{O}(DMN\log(N)T)$). On the other hand, the training time required for AdaBoost.MH was 20 to 40 times larger than for ConfAdaBoost.M1 on the larger datasets (*e.g.* Letter or Pendigits). Therefore, training AdaBoost.MH is not feasible for extremely large datasets.

5 Evaluation on the PAMAP2 dataset

The PAMAP2 dataset is a physical activity monitoring dataset created and released recently [28, 29], and is included in the UCI machine learning repository as well. The dataset was recorded from 18 physical activities performed by 9 subjects, wearing 3 *inertial measurement units* (IMU) and a heart rate monitor. Each of the subjects followed a predefined data collection protocol of 12 activities (lie, sit, stand, walk, run, cycle, Nordic walk, iron, vacuum clean, rope jump, ascend and descend stairs), and optionally performed a few other activities (watch TV, computer work, drive car, fold laundry, clean house, play soccer). Therefore, the PAMAP2 dataset not only includes basic physical activities and postures, but also a wide range of everyday, household and fitness activities. A more detailed description of the dataset can be found in [28]. In this section first an activity recognition and an intensity estimation classification problem is defined on the PAMAP2 dataset. These classification tasks are described in detail, highlighting also the differences to the UCI benchmark datasets of the previous section and pointing out the special challenge these problems pose. Using the defined classification tasks different boosting methods are evaluated and compared to the proposed ConfAdaBoost.M1 algorithm.

5.1 Definition of the classification problems

The benchmark of [28, 29] defined 4 different classification problems on the PAMAP2 dataset. One of these problems — called *All activity recognition task* — uses the 12 activities of the data collection protocol, defining 12 classes corre-

sponding to the activities. This classification task is extended in this paper with 3 additional activities from the optional activity list: fold laundry, clean house and play soccer.[1] This activity recognition task of 15 different activity classes will be referred to as the 'PAMAP2_AR' task throughout this work. Moreover, an intensity estimation classification task is defined on the PAMAP2 dataset: using all 18 activities, the goal is to distinguish activities of light, moderate and vigorous effort (referred to as 'PAMAP2_IE' task). The ground truth for this rough intensity estimation task is based on the *metabolic equivalent* (MET) of the different physical activities, provided by [1]. Therefore, the 3 intensity classes are defined as follows: lie, sit, stand, drive car, iron, fold laundry, clean house, watch TV and computer work are regarded as activities of light effort (< 3.0 METs); walk, cycle, descend stairs, vacuum clean and Nordic walk as activities of moderate effort (3.0-6.0 METs); run, ascend stairs, rope jump and play soccer as activities of vigorous effort (> 6.0 METs).

Contrary to the 8 UCI benchmark datasets used for the experiments in the previous section, the PAMAP2 dataset does not directly provide a feature vector with each of the instances, but only provides timestamped raw sensory data from the 3 IMUs and the heart rate monitor. Therefore, the raw signal data needs to be processed first in order to be used by classification algorithms. A *data processing chain* (DPC) is applied on the raw sensory data including preprocessing, segmentation, feature extraction and classification steps, *cf.* Fig. 3. In the preprocessing step, the timestamped raw acceleration[2] and heart rate data is synchronised, and possible wireless data loss compensated with linear interpolation. To obtain at least 2–3 periods from the periodic movement of various aerobic activities (walking, cycling, *etc.*), and to as-

---

[1] The remaining 3 activities from the dataset are discarded for the following reasons: *drive car* contains data from only one subject, while *watch TV* and *computer work* are not considered due to their high resemblance to the *sit* class.

[2] Previous work shows (*e.g.* in [21]), that both for activity recognition and for intensity estimation, accelerometers outperform gyroscopes. Therefore, from all 3 IMUs, only data from the accelerometers is used in the subsequent data processing steps.
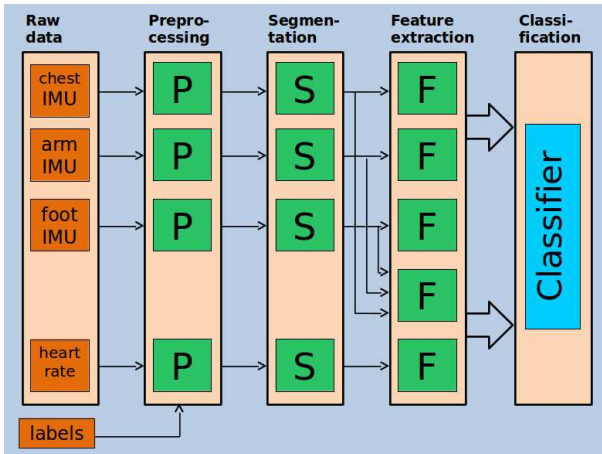
Fig. 3 The data processing chain applied on the PAMAP2 dataset.

sure effective FFT-calculation, a window size of 512 samples (equivalent with 5.12 seconds due to the sampling rate of 100 Hz) is selected for the segmentation step. Therefore, the preprocessed data is segmented using a sliding window with the defined 5.12 seconds of window size, shifted by 1 second between consecutive windows. From each window segment, a total of 137 features are extracted: 133 features from IMU acceleration data (such as mean, standard deviation, energy, entropy, correlation, *etc.*) and 4 features from heart rate data (mean and gradient). These extracted features serve as input to the classification step, in which different boosting algorithms are evaluated. The data processing steps preprocessing, segmentation and feature extraction are further described in [28].

The main parameters of the PAMAP2 classification tasks are summarised in Table 1. It is clear that, compared to the other datasets of Table 1, the classification problems defined on the PAMAP2 dataset are significantly more complex, considering the number of instances and especially the number of variables. To get a first impression about the difficulty of these tasks, experiments with a C4.5 decision tree classifier are performed: an error rate of 34.21% is achieved on the PAMAP2_AR and 11.02% on the PAMAP2_IE task, averaged over 10 test runs. This result serves as baseline performance, showing that improvement is required and to be expected while applying different boosting methods.

The experiments presented below in this section compare the newly introduced ConfAdaBoost.M1 algorithm to the boosting methods AdaBoost.M1, QuinlanAdaBoost.M1 and SAMME. The selection of these algorithms for comparison was already explained in Section 4.1. The comparison to AdaBoost.MH is not considered here due to the unfeasible training time it would require, given the complexity of the classification tasks and that the actual size of the training set is a multiple of that of the other algorithms (*cf.* also the discussion in Section 4.2). Similar to the previous section, the C4.5 decision tree classifier is used for each of the

boosting algorithm as weak learner. An important difference in the realization of the experiments in this section is the applied evaluation technique. As discussed *e.g.* in [27,31], a subject independent validation technique simulates best the goals of systems and applications using physical activity recognition. Therefore, *leave-one-subject-out* (LOSO) 9-fold cross-validation is used in this section, while evaluating each method from 1 up to 500 boosting iterations.

5.2 Results and discussion

The averaged results of the 10 test runs on the PAMAP2 classification tasks are shown in Fig. 4, the test error rates of the 4 evaluated boosting methods are included in Table 2. Compared to the baseline accuracy of the decision tree classifier, all boosting methods significantly improve the performance. The ConfAdaBoost.M1 algorithm clearly outperforms the other methods: *e.g.* on the PAMAP2_AR task, compared to the performance of the second best SAMME algorithm a reduction of the test error rate by nearly 20% can be observed. This reduction of the test error rate is statistically significant with a p-value smaller than 0.001. As discussed in Section 4.2, it was expected that the most significant improvement from all the datasets evaluated in this work is achieved on the PAMAP2_AR classification task, since it represents the largest and most complex classification problem.

Similar to the results of Fig. 1 and 2, the algorithms AdaBoost.M1 and QuinlanAdaBoost.M1 reach the stopping criterion with fewer boosting iterations. However, contrary to the results of the previous section, QuinlanAdaBoost.M1 performs significantly better here (especially on the PAMAP2_AR task), confirming that it is even worth to apply the confidence-based modification to only the prediction step of the original AdaBoost.M1 algorithm, as proposed in [25]. However, compared to QuinlanAdaBoost.M1, ConfAdaBoost.M1 reduces the test error rate by 20%. Therefore, the major part of the performance improvement achieved by ConfAdaBoost.M1 comes from the confidence-based extension of both the training and prediction step of the original AdaBoost.M1 algorithm, as also confirmed by the results on the 8 other UCI datasets. Therefore ConfAdaBoost.M1 is clearly a significant improvement over QuinlanAdaBoost.M1.

ConfAdaBoost.M1 also adopts one of the beneficial characteristics of boosting: it rarely overfits a classification problem. The only result indicating overfitting is on the PAMAP2_AR task, the reasons need further investigation. However, it should be noted that generally the behaviour of AdaBoost algorithms concerning overfitting is a controversial topic in the scientific community, *cf. e.g.* [20]. Moreover, different techniques have been proposed to avoid overfitting even in rare cases, *e.g.* by implementing data-derived
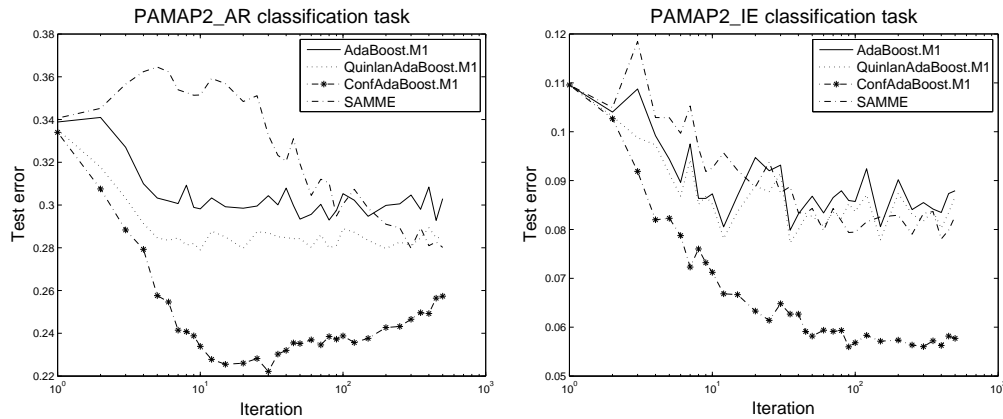
Fig. 4 Test error of the 4 evaluated boosting algorithms on the PAMAP2 classification tasks. The results are averages over 10 test runs.

early stopping rules [43]. Nevertheless, in the example presented in Fig. 4, the performance of ConfAdaBoost.M1 on the PAMAP2_AR task is significantly better than that of the other evaluated boosting methods even with larger boosting iterations.

To better understand the results of this section, the confusion matrix of the best performing classifier (ConfAda-Boost.M1 with 30 boosting iterations) on the PAMAP2_AR task is presented in Table 3.[3] The numbering of the activities in the table corresponds to the activity IDs as given in the PAMAP2 dataset. The results are averaged over 10 test runs, the overall accuracy is 77.78%. The confusion matrix shows that some activities are recognised with high accuracy, *e.g.* lie, walk or even distinguishing between ascend and descend stairs. Misclassifications in Table 3 have several reasons. For example, the over 5% confusion between sit and stand can be explained with the positioning of the sensors: an IMU on the thigh would be needed for a reliable differentiation of these postures. Moreover, ironing has a similar characteristics from the used set of sensors' point of view, especially compared to talking and gesticulating during standing. Another example of overlapping activity characteristics comes from the introduction of playing soccer into this classification problem. Playing soccer is a composite activity, and it is for instance not trivial to distinguish running with a ball from just running. The significant confusion between the different household activities (vacuum clean, iron, fold laundry and clean house — the latter mainly consisting of dusting shelves) indicates that they can not be reliably distinguished with the given set of sensors. However, arguably, the main reason for the misclassifications in Table 3 is the diversity in how subjects perform physical activities. Therefore,

to further increase the accuracy of physical activity recognition, personalisation approaches should be introduced and investigated.

6 Feasibility of ConfAdaBoost.M1 for mobile systems

The previous sections introduced and evaluated a novel confidence-based boosting algorithm. It was shown that Conf-AdaBoost.M1 significantly outperforms existing, commonly used methods. However, the feasibility of the proposed algorithm for online mobile applications remains an open question. As pointed out in Section 1, boosted decision tree classifiers have a simple structure and are thus easy to implement. However, boosting is complex in the way that the size of the classifier is about T-times larger (T being the number of boosting iterations) than the applied base-level classifier. This means that boosted classifiers have larger computational requirements. Moreover, as shown by the results in Sections 4.2 and 5.2, more boosting iterations can be reached during training with ConfAdaBoost.M1 compared to *e.g.* AdaBoost.M1, raising further the computational requirements. On the other hand, the mobile systems where physical activity monitoring applications are usually running on have a restriction on available computational power. Therefore, this section describes two empirical studies performed to examine the feasibility of the proposed ConfAdaBoost.M1 algorithm for such mobile systems. The first study, presented in Section 6.1, investigates the feasibility for an online general activity monitoring model. The second study, presented in Section 6.2, applies ConfAdaBoost.M1 as part of a recently introduced personalisation concept for physical activity recognition.

Both empirical studies are carried out with a state-of-the-art mobile system prototype. This prototype consists of Shimmer[4] wearable wireless sensors, a wireless heart rate

---

[3] Recently new error metrics were introduced for continuous activity recognition, *e.g.* insertion, merge, overfill, *etc.* [39,41]. However, contrary to activity recognition in *e.g.* home or industrial settings, for physical activity monitoring the frame by frame metrics (precision, recall, F-measure and accuracy: all derivable from the confusion matrix) are sufficient, as discussed in [28].

[4] http://www.shimmersensing.com

Table 3 Confusion matrix of the PAMAP2_AR classification task using the ConfAdaBoost.M1 classifier and 30 boosting iterations. The table shows how different annotated activities are classified in [%].

Annotated activity	Recognised activity														
	1	2	3	4	5	6	7	12	13	16	17	18	19	20	24
1 lie	97.1	1.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.9	0.0	0.0
2 sit	2.0	84.8	5.4	0.0	0.0	0.5	0.0	0.0	0.0	0.1	4.1	0.6	2.5	0.0	0.0
3 stand	0.0	6.0	83.0	0.0	0.0	0.0	0.0	0.0	0.0	0.3	7.4	0.9	2.4	0.0	0.0
4 walk	0.0	0.0	0.0	92.2	0.0	0.0	0.5	6.8	0.0	0.0	0.0	0.0	0.0	0.4	0.0
5 run	0.0	0.0	0.0	0.0	89.9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	10.1	0.0
6 cycle	0.0	0.0	0.0	1.1	0.0	91.7	0.4	0.5	0.0	1.3	0.1	0.0	4.9	0.0	0.0
7 Nordic walk	0.0	0.0	0.0	2.7	0.0	0.0	89.1	1.1	0.1	0.0	0.0	0.0	0.1	7.0	0.0
12 asc. stairs	0.0	0.0	0.0	6.4	0.0	0.2	0.3	87.2	2.6	0.7	0.0	0.0	0.3	2.5	0.0
13 desc. stairs	0.0	0.0	0.0	0.1	0.1	0.0	0.2	6.7	91.3	0.0	0.0	0.0	0.2	0.7	0.7
16 vacuum clean	0.0	0.0	0.1	0.0	0.0	1.1	0.0	0.3	0.4	73.5	1.3	0.3	23.1	0.0	0.0
17 iron	0.0	2.6	0.8	0.0	0.0	0.0	0.0	0.0	0.0	1.2	77.7	5.0	12.7	0.0	0.0
18 fold laundry	0.0	1.1	1.5	0.0	0.0	0.1	0.0	0.0	0.0	8.9	61.1	11.1	16.2	0.0	0.0
19 clean house	0.5	0.6	3.4	0.0	0.0	1.7	0.0	1.2	0.7	21.4	18.1	5.0	47.4	0.0	0.0
20 play soccer	0.0	0.0	0.0	5.1	27.6	1.4	2.8	7.3	20.6	1.7	0.0	0.0	0.1	20.7	12.8
24 rope jump	0.0	0.0	0.0	0.0	32.0	0.0	0.0	1.3	0.1	0.0	0.0	0.0	0.0	7.8	58.8

Table 4 Feasibility study I: online physical activity monitoring. Comparing computational cost and performance of the C4.5 decision tree, AdaBoost.M1 and ConfAdaBoost.M1 classifiers on the Samsung Galaxy S III smartphone.

Classifier	Computation time (ms)		Accuracy [%]	
	average	maximum	intensity estimation	activity recognition
C4.5 decision tree	3.32	42	94.36	93.84
AdaBoost.M1	24.54	131	96.18	98.98
ConfAdaBoost.M1	51.54	150	99.79	100

monitor and an Android smartphone. Three Shimmer 2r baseboard units (containing each an accelerometer) are placed on chest, lower arm and ankle positions, a sensor setup already used for the PAMAP2 dataset recording [28, 29]. Moreover, a Zephyr Bioharness 3[5] heart rate monitor is included in the prototype. Finally, as mobile control unit, a Samsung Galaxy S III (which contains a 1.4 GHz quad-core Cortex-A9 CPU and 1 GB of RAM) Android smartphone is used.

6.1 Study I: Online general physical activity monitoring

The goal of this study is to show the feasibility of the ConfAdaBoost.M1 algorithm for online physical activity monitoring on mobile devices. For this purpose the entire data processing chain of Fig. 3 is implemented in `Java` on the Samsung Galaxy S III smartphone. The structure of the implementation includes a data collection thread (including pre-processing and segmentation of raw sensory data) for each of the sensors, and a data processing thread for feature extraction and classification. The feature extraction step is optimised in a way that each feature should be computed at most once each window segment. As for the classification step, ConfAdaBoost.M1 is compared to a C4.5 decision tree classifier and to AdaBoost.M1. Both boosting classifiers have the C4.5 decision tree as weak learner, and the T number of boosting iterations is set to 100. Both intensity estimation

and activity recognition are included in this mobile physical activity monitoring application, as defined in Section 5.1.

The protocol of this empirical study can be described as follows. First, data is recorded from two subjects performing various activities while wearing the mobile system. This data is used for training all classifiers offline, for both the intensity estimation and the activity recognition task. Then, with each of the trained classifier an approximately 20 minutes protocol is carried out by one of the subjects. The same protocol is followed with each classifier, including the following wide range of activities: lying, sitting, standing, walking, running, cycling, ascending and descending stairs.

Table 4 shows the comparison of the three different decision tree-based classifiers. The computation time includes the classification and the computation of the required features for the respective classification step, and was computed in the above mentioned data processing thread of the online application. It is clear that even the maximum computation time of each classifier is far below the restriction of 1 second. This restriction is defined by the fact that the segmentation step of the DPC uses a sliding window shifted by 1 second, thus the data processing thread has maximum 1 second for each processing step. The difference between AdaBoost.M1 and ConfAdaBoost.M1 in computation time can be explained by the fact that the training of the AdaBoost.M1 algorithm stops at an earlier boosting round, thus this classifier is of smaller size. With the ConfAdaBoost.M1 algorithm on the other hand, the predefined iteration num-

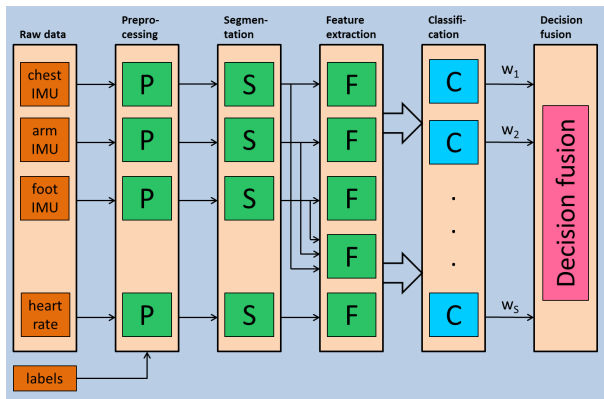---
[5] http://www.zephyranywhere.com

Fig. 5 Data processing chain: a novel concept of personalisation.

ber of 100 is reached during the training for both intensity estimation and activity recognition.

Overall, this empirical study proves the feasibility of using ConfAdaBoost.M1 for online activity monitoring on smartphones, there are no limitations considering the computational costs. Moreover, the comparison of the three classifier's performance in Table 4 confirms previous results of this work: The ConfAdaBoost.M1 algorithm outperforms the other classifiers on both the intensity estimation and the activity recognition task.

6.2 Study II: Personalised physical activity recognition

Personalisation of physical activity recognition has become a topic of interest recently. It is motivated by the fact that activity monitoring systems are usually trained on a large number of subjects, and then used by a new subject from whom data is not available in the training phase. Most existing personalisation approaches focus on the classification step of the DPC. A common personalisation concept is to adapt the parameters of a previously trained general model to the new user. However, the drawback of changing the parameters of a general model is that either the model is simple (*e.g.* the decision tree classifiers used in [22,44]) and thus only low performance can be expected on more challenging activity recognition tasks, or the general model is complex and thus resulting in unfeasible computational costs for mobile applications [4].

A novel general concept of personalisation has been introduced recently in [33]. In this concept, personalisation is applied in the decision fusion step of the DPC, as shown in Fig. 5. The general model consists of a set of S classifiers (experts), all weighted the same: $w_i = 1$, $i = 1, \ldots, S$. Using new labeled data from a previously unknown subject, only the w_i weights of the experts are retrained, the classifiers themselves remain the same. In order to show that this concept is a valid approach for personalisation, different methods based on the idea of weighted majority voting

have successfully been applied to increase the performance of the general model for new individuals: *weighted majority algorithm* (WMA), *randomised weighted majority algorithm* (RWMA), *weighted majority voting* (WMV) and the newly introduced *dependent experts* (DE) algorithm [33]. The main idea of the latter is that the confidence of an expert's prediction depends on the decision of all other experts. Therefore, the training part of the DE algorithm results in a matrix of weights, containing weights for each of the experts when the majority vote of all other experts is a given class. For further details the reader is referred to [33].

One of the key goals and benefits of the above described personalisation concept is that a fast personalisation of even advanced classifiers is enabled. This way the personalisation concept can handle complex activity recognition tasks (*e.g.* the recognition of not only a few basic, but a large number of physical activities), while still feasible for mobile applications regarding its computational complexity [33]. It is expected thereby that the new user — after recording new labeled data — receives the personalised model within a short time. In order to investigate this an empirical study is designed and carried out, implementing the DPC of Fig. 5 on the Samsung Galaxy S III smartphone.

The procedure of the empirical study can be described as follows. First, data is recorded from one subject (wearing the previously described mobile system prototype) during 6 sessions, each session including the following 7 activities: lying, sitting, standing, walking, running, ascending and descending stairs. These recordings are used to create the general model, consisting of 6 classifiers (each of these classifiers is trained using data from one of the sessions). C4.5 decision tree, AdaBoost.M1 and ConfAdaBoost.M1 (both with decision tree as base-level classifier) are used and compared as classifiers in the general model. Then, labeled data from a second subject is recorded, while performing each of the 7 activities for approximately one minute (this parameter has been determined in [33], as trade-off between classification accuracy and the time required for new data recording). This new data is used to retrain the weights of the 6 classifiers in the general model. The retraining of the weights is performed directly on the smartphone, for each of the 4 analysed algorithms. For each classifier — personalisation algorithm pair the retraining is run 5 times, results will present the average of these test runs.

Table 5 presents the average retraining time of each of the weighted majority voting algorithms and each type of classifier. The interpretation of these results is the following: After the second subject recorded the required new training data and started the retraining of the general model on the smartphone, how long did he have to wait to receive his personalised model. For each of the retraining algorithms, the major computational time is spent to predict the label of each new sample by each of the general model's classifier

Table 5 Feasibility study II: personalisation of physical activity recognition. Computational time [s] required for the retraining of the general model. Decision tree, AdaBoost.M1 and ConfAdaBoost.M1 are each tested as classifiers used in the general model. The proposed personalisation approach is evaluated with the weighted majority voting-based methods WMV, WMA, RWMA and DE.

Classifier	WMV	WMA	RWMA	DE
C4.5 decision tree	4.01	3.89	4.01	4.05
AdaBoost.M1	10.91	11.03	10.82	10.84
ConfAdaBoost.M1	30.89	30.48	31.01	31.00

(for which the by far most computationally intensive part is the feature calculation in the data processing chain). With an effective implementation this has to be done exactly once for each sample — classifier pair, even when applying the DE algorithm. Therefore, the retraining time for all 4 algorithms is expected to be similar, as confirmed by the results of Table 5. Furthermore, the retraining of the general model when consisting of ConfAdaBoost.M1 classifiers takes the longest, since this classifier is the most complex, thus includes the calculation of the most features. Nevertheless, the retraining time of approximately 30 seconds is still acceptable: The new user receives a complex personalised system after only waiting half a minute. Therefore, this empirical study proves the feasibility of using ConfAdaBoost.M1 for mobile, personalised physical activity monitoring as well.

7 Conclusion

This paper presented a confidence-based extension of the well-known AdaBoost.M1 algorithm, called ConfAdaBoost.M1. The new algorithm builds on established ideas of existing boosting methods, combining some of their benefits. The ConfAdaBoost.M1 algorithm has been evaluated on various benchmark datasets, comparing it to the most commonly used boosting techniques. ConfAdaBoost.M1 performed significantly best among these algorithms, especially on the larger and more complex activity monitoring problems: on the PAMAP2_AR task the test error rate was reduced by nearly 20% compared to the second best performing classifier. Therefore, the main motivation of proposing this new boosting variant — namely to overcome some of the challenges defined by recent benchmark results in physical activity monitoring — was achieved successfully. Moreover, empirical studies proved the feasibility of using the ConfAdaBoost.M1 algorithm for physical activity monitoring applications in mobile systems, there are no limitations considering the computational costs.

This work presented experimental proof on various datasets in different application areas that the ConfAdaBoost.M1 algorithm is superior to existing methods, and using it improves on classification performance. The main concepts of the new method are clear and comprehensible, but a theoretical interpretation of the algorithm and explanation of its success remains for future work. Moreover, it is planned to slightly modify ConfAdaBoost.M1 — similar to *e.g.* the

modification proposed by SAMME over the original AdaBoost.M1 algorithm — to loosen the stopping criterion of $e_t \geq 0.5$, thus allowing the usage of "weak" weak learners (such as decision stumps). However, boosting decision trees proved to be very successful in the experiments presented in this work, and will remain (due to its many benefits discussed in this paper) one of the most widely used classifiers especially in the field of physical activity monitoring.

References

1. Ainsworth, B. E., Haskell, W. L., Whitt, M. C., Irwin, M. L., Swartz, a. M., Strath, S. J., O'Brien, W. L., Bassett, D. R., Schmitz, K. H., Emplaincourt, P. O., Jacobs, D. R., and Leon, a. S. Compendium of physical activities: an update of activity codes and MET intensities. *Medicine and Science in Sports and Exercise*, 32(9), pp. 498–504, September 2000.
2. Alimoglu, F., and Alpaydin, E. Methods of combining multiple classifiers based on different representations for ben-based handwritten digit recognition. In *Proceedings of 5th Turkish Artificial Intelligence and Artificial Neural Networks Symposium (TAINN)*, Istanbul, Turkey, 1996.
3. Bache, K., and Lichman, M. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2013. http://archive.ics.uci.edu/ml.
4. Berchtold, M., Budde, M., Gordon, D., Schmidtke, H., and Beigl, M. ActiServ: Activity recognition service for mobile phones. In *Proceedings of IEEE 14th International Symposium on Wearable Computers (ISWC)*, Seoul, South Korea, October 2010.
5. Eibl, G., and Pfeiffer, K. P. How to make AdaBoost.M1 work for weak base classifiers by changing only one line of the code. In *Proceedings of 13th European Conference on Machine Learning (ECML)*, pp. 72–83, Helsinki, Finland, August, 2002.
6. Ermes, M., Pärkkä, J., and Cluitmans, L. Advancing from offline to online activity recognition with wearable sensors. In *Proceedings of 30th Annual International IEEE EMBS Conference*, pp. 4451–4454, Vancouver, BC, Canada, August 2008.
7. Faddoul, J. B., Chidlovskii, B., Gilleron, R., and Torre, F. Learning multiple tasks with boosted decision trees. In *Proceedings of 2012 European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML-PKDD)*, pp. 681–696, Bristol, UK, September 2012.
8. Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7, pp. 179–188, 1936.
9. Freund, Y. The strength of weak learnability. *Information and Computation*, 121(2), pp. 256–285, September 1995.

10. Freund, Y., and Schapire, R. E. A desicion-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), pp. 119–139, August 1997.

11. Frey, P. W., and Slate, D. J. Letter recognition using Holland-style adaptive classifiers. *Machine Learning*, 6(2), pp. 161–182, March 1991.

12. Friedman, J., Hastie, T., and Tibshirani, R. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2), pp. 337–407, 2000.

13. Gómez-Verdejo, V., Ortega-Moral, M., Arenas-García, J., and Figueiras-Vidal, A. R. Boosting by weighting critical and erroneous samples. *Neurocomputing*, 69(7-9), pp. 679–685, March 2006.

14. Haskell, W. L., Lee, I.-M., Pate, R. R., Powell, K. E., Blair, S. N., Franklin, B. A., Macera, C. A., Heath, G. W., Thompson, P. D., Bauman, A. Physical activity and public health: Updated recommendation for adults from the American College of Sports Medicine and the American Heart Association. *Medicine and Science in Sports and Exercise*, 39(8), pp. 1423–1434, August 2007.

15. Huang, J., Ertekin, S., Song, Y., Zha, H., and Giles, C. L. Efficient multiclass boosting classification with active learning. In *Proceedings of SIAM International Conference on Data Mining (SDM)*, Minneapolis, MN, USA, April 2007.

16. Jin, X., Hou, X., and Liu, C.-L. Multi-class AdaBoost with hypothesis margin. In *Proceedings of 20th International Conference on Pattern Recognition (ICPR)*, pp. 65–68, Washinton, DC, USA, August, 2010.

17. Kuncheva, L. I. *Combining pattern classifiers: methods and algorithms*. Wiley-Interscience, 2004.

18. Lee, M.-h., Kim, J., Kim, K., Lee, I., Jee, S. H., Yoo, S. K. Physical activity recognition using a single tri-axis accelerometer. In *Proceedings of World Congress on Engineering and Computer Science (WCECS)*, San Francisco, CA, USA, October 2009.

19. Long, X., Yin, B., and Aarts, R. M. Single-accelerometer based daily physical activity classification. In *Proceedings of 31st Annual International IEEE EMBS Conference*, pp. 6107–6110, Minneapolis, MN, USA, September 2009.

20. Mease, D., and Wyner, A. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9(Feb), pp. 131–156, 2008.

21. J. Pärkkä, M. Ermes, K. Antila, M. van Gils, A. Mänttäri, and H. Nieminen. Estimating intensity of physical activity: a comparison of wearable accelerometer and gyro sensors and 3 sensor locations. In *Proceedings of 29th Annual International IEEE EMBS Conference*, pp. 1511–1514, Lyon, France, August 2007.

22. Pärkkä, J., Cluitmans, L., and Ermes, M. Personalization algorithm for real-time activity recognition using PDA, wireless motion bands, and binary decision tree. *IEEE Transactions on Information Technology in Biomedicine*, 14(5), pp. 1211–1215, September 2010.

23. Quinlan, J. R., Compton, P. J., Horn, K. A., and Lazarus, L. Inductive knowledge acquisition: a case study. In *Proceedings of 2nd Australian Conference on Applications of Expert Systems*, pp. 137–156, Sydney, Australia, May 1986.

24. Quinlan, J. R. *C4.5: programs for machine learning*. San Mateo: Morgan Kaufmann, 1993.

25. Quinlan, J. R. Bagging, boosting and C4.5. In *Proceedings of 13th National Conference on Artificial Intelligence (AAAI)*, pp. 725–730, Portland, OR, USA, August 1996.

26. Ravi, N., Dandekar, N., Mysore, P., and Littman, M. Activity recognition from accelerometer data. In *Proceedings of 17th Conference on Innovative Applications of Artificial Intelligence (IAAI)*, pp. 1541–1546, Pittsburgh, PA, USA, July 2005.

27. Reiss, A., Weber, M., and Stricker, D. Exploring and extending the boundaries of physical activity recognition. In *Proceedings of 2011 IEEE International Conference on Systems, Man and Cybernetics (SMC), Workshop on Robust Machine Learning Techniques for Human Activity Recognition*, pp. 46–50, Anchorage, AK, USA, October 2011.

28. Reiss, A., and Stricker, D. Creating and benchmarking a new dataset for physical activity monitoring. In *Proceedings of 5th Workshop on Affect and Behaviour Related Assistance (ABRA)*, Crete, Greece, June 2012.

29. Reiss, A., and Stricker, D. Introducing a new benchmarked dataset for activity monitoring. In *Proceedings of IEEE 16th International Symposium on Wearable Computing (ISWC)*, pp. 108–109, Newcastle, UK, June 2012.

30. Reiss, A., and Stricker, D. Aerobic activity monitoring: towards a long-term approach. *International Journal of Universal Access in the Information Society (UAIS)*, March 2013.

31. Reiss, A., Hendeby, G., and Stricker, D. Towards robust activity recognition for everyday life: methods and evaluation. In *Proceedings of 7th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, Venice, Italy, May 2013.

32. Reiss, A., Hendeby, G., and Stricker, D. Confidence-based multiclass AdaBoost for physical activity monitoring. In *Proceedings of IEEE 17th International Symposium on Wearable Computing (ISWC)*, pp. 13–20, Zurich, Switzerland, September 2013.

33. Reiss, A., and Stricker, D. Personalized mobile physical activity recognition. In *Proceedings of IEEE 17th International Symposium on Wearable Computing (ISWC)*, pp. 25–28, Zurich, Switzerland, September 2013.

34. Schapire, R. E. The strength of weak learnability. *Machine Learning*, 5(2), pp. 197–227, June 1990.

35. Schapire, R. E. Using output codes to boost multiclass learning problems. In *Proceedings of 14th International Conference on Machine Learning (ICML)*, pp. 313–321, Nashville, TN, USA, July 1997.

36. Schapire, R. E., and Singer, Y. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3), pp. 297–336, December 1999.

37. Siebert, J. P. Vehicle recognition using rule based methods. Turing Institute (Glasgow, Scotland), 1987.

38. Tapia, E. M., Intille, S. S., Haskell, W., Larson, K., Wright, J., King, A., Friedman, R. Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor. In *Proceedings of IEEE 11th International Symposium on Wearable Computing (ISWC)*, pp. 1–4, Boston, MA, USA, October 2007.

39. van Kasteren, T., Alemdar, H., and Ersoy, C. Effective performance metrics for evaluating activity recognition methods. In *Proceedings of 24th International Conference on Architecture of Computing Systems (ARCS)*, Como, Italy, February 2011.

40. Vezhnevets, A., and Vezhnevets, V. Modest AdaBoost - teaching AdaBoost to generalize better. In *Proceedings of 15th International Conference on Computer Graphics and Applications (Graphicon)*, Novosibirsk, Russia, June 2005.

41. Ward, J. A., Lukowicz, P., and Gellersen, H. W. Performance metrics for activity recognition. *ACM Transactions on Intelligent Systems and Technology*, 2(1), January 2011. Article No. 6.

42. Wu, X., Kumar, V., Quinlan, J. R., Ghosh, J., Yang, Q., Motoda, H., McLachlan, G. J., Ng, A., Liu, B., Yu, P. S., Zhou, Z., Steinbach, M., Hand, D. J., and Steinber, D. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1), pp. 1–37, December 2007.

43. Zhang, T., and Yu, B. Boosting with early stopping: Convergence and consistency. *The Annals of Statistics*, 33(4), pp. 1538–1579, 2005.

44. Zhao, Z., Chen, Y., Liu, J., Shen, Z., and Liu, M. Cross-people mobile-phone based activity recognition. In *Proceedings of 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pp.2545–2550, Barcelona, Spain, July 2011.

45. Zhu, J., Rosset, S., Zou, H., and Hastie, T. Multi-class Adaboost. *Technical Report 430, Department of Statistics, University of Michigan*, 2005.

46. Zhu, J., Zou, H., Rosset, S., and Hastie, T. Multi-class Adaboost. *Statistics and Its Interface*, 2, pp. 349–360, 2009.