Human Activity Recognition by machine learning methods

Ole M. Brastein [1], Roland Olsson [2], Nils-Olav Skeie [1] and Thomas Lindblad [3]

[1] Høgskolen I Sørøst Norge, Porsgrunn
[2] Høgskolen I Østfold
[3] Kungliga Tekniska Högskolan, Stockholm

Abstract

Recognition of human activity from sensor data is a research field of great potential. Giving autonomous systems the ability to identify what a human subject is doing at a given time is highly useful in many industries, particularly in health care and security monitoring. Our results, using a public domain dataset, show that the state-of-the-art decision tree ensemble algorithm XGBoost gives an accuracy of 94.6% validated on an independent test set. Previously published results using support vector machines (SVM) gave an accuracy of 90.2%. As far as we know, our result is the new state of the art for this data set. Recognition of human activity carries potential privacy concerns, which to some degree constrain the choice of sensor technology. Therefore, systems such as ours which can identify activities from simple inertial sensors, e.g. accelerators and gyroscopes are of particular interest. Data from such inertial sensors are difficult to interpret using mechanistic models; hence the field of Machine Learning is particularly interesting for this application.

1 Introduction

Human Activity Recognition (HAR) is the process of identifying what a person is doing based on sensor readings. Activities are generally divided into classes, and the goal of HAR is to identify which class of activity is performed. There are many applications of HAR. For example, a HAR system could detect abnormal activity in a crowd of people and allow identification of possible threatening situations, or detect that a person is in need of assistance. Another situation where identifying human activity can be very useful is the monitoring of elderly people in need of care [1].

Several types of sensor systems can be used to classify human activities. One popular approach is computer vision systems. Another applicable sensor system is the use of inertial sensors, e.g. accelerometers and gyroscopes. Most modern smartphones have such sensors built into them. Modern silicon fabrication technology has allowed micro-scale sensors, often referred to as Micro-Electro-Mechanical Systems (MEMS) such as accelerometers and gyroscopes to be produced cheaply and in small packages. This makes the identification of human activity from inertial data especially interesting, as such sensor systems could be applied without any practical inconvenience to the wearer of the sensors.

The data set used in this project was taken from J. L. Reyes-Ortiz, et. al. [2], and has been used in several projects [3-5]. The authors use machine learning methods to analyze the data. Support vector machines (SVM) are used in [3] to classify six different activities, walking, walking upstairs, walking downstairs, standing, sitting and laying. Classification results are on the order of 70-90% with the laying activity having a particularly high success rate of 100% correct classifications.

Based on the promising results in [3] it is of interest to evaluate other classification algorithms and compare their performance with the SVM results. In the present work, the same data set as used in [3] is analyzed with three decision tree classification methods, namely C5.0, Random Forest and XGBoost [6].

2 Methods and algorithms

XGBoost is generally the most accurate machine learning (ML) method in use today, and it is used to win around 50% of Kaggle competitions. XGBoost is a state of the art tree ensemble method, which contains an extensive set of regularization mechanisms to prevent overfitting. Random Forest (RF) is by far the most used tree ensemble method; hence it is of interest to compare results from Random Forest and XGBoost. The C5.0 algorithm with boosting is included here only for comparison with the more modern methods; XGBoost and RF.

2.1 Model development in machine learning

Models of physical systems can be classified by two different types. Mechanistic models are based on physical equations, such as mass and energy balances, while empirical models are based primarily on measurement data. Machine learning (ML) uses predominantly empirical models [7]. Examples of typical empirical models used in ML include artificial neural networks (ANN) and decision tree models [7-9]. Empirical models can be used for regression, where the output is a numerical value, or for classification giving an output from a set of discrete classes. In this paper all the applied methods are based on decision trees for classification. Most ML methods using decision trees include multiple trees in the form of an ensemble or forest. Different methods train and combine trees in different ways.

Empirical models are calibrated based on a set of training data. To verify that the model gives reasonable predictions of the physical system, a second test data set must be used. The testset must be independent of the training data in order to ensure realistic predictions of model performance. Cross-validation (CV) [7] can be used to create pseudo-independent datasets during model calibration in order to perform intermediate tests for the purpose of e.g. finding optimal hyper parameters. A hyper parameter, also called a tuning parameter, is a parameter of the method itself, and not subject to calibration in the training of a model. Examples of hyper parameters can be the maximum depth of a decision tree or the number of nodes in an ANN. Much of this paper is concerned with the discussion of choosing hyper parameters, i.e. tuning of the method, such that the calibrated model has optimal classification accuracy.

A decision tree is based on a set of splitting criteria where each criterion constitutes a node in the tree. Each possible outcome of a split constitutes a branch in the tree, such that each branch represents a segment of the data. After a node, each branch leads either to a new splitting node or to a leaf node. A leaf node in a classification tree contains a single class which constitutes the model output. The models prediction for a new sample is determined by the output class associated with the leaf node the sample reaches after travelling through the various splits in the tree.

At each node a single predictor, i.e. a variable in the data set, is used to separate the data into segments [7]. The number of segments and the value of the splitting criteria depend on the type of predictor. For continuously valued predictors as used in this paper, each tree node splits the data in two parts based on a single value. This value is chosen such that the two sub-sets of the data are optimizing a chosen performance criteria such as Gini impurity index [7] or Shannon entropy [7]. For a classification tree the chosen performance criteria describes quantitatively the degree of success for a specific split in segregating the different classes. Optimal performance indicates that the different classes are segregated by the split to the highest degree possible.

To avoid overfitting, i.e. making a model that describes details in the training data that are essentially random noise, regularisation is applied. Different methods use different selection of regularisation methods.

2.2 Random Forest

Random Forest [10] is based on the idea of using bootstrapping, i.e. random selection of training samples with replacement. Furthermore, Random Forest extends the idea of randomization by also choosing a selection of predictors at random. If all the same predictors are used in every bootstrap aggregated, i.e. “bagged”, iteration, the resulting trees will have some degree of correlation induced by the predictors. However, by randomly selecting both predictors and samples, Random Forest decorrelates the trees in the ensemble further. The number of predictors chosen at each iteration is often denominated m_{try}, and is considered a hyper parameter of the algorithm. Additionally, the number of trees, i.e. bagging iterations, must also be chosen [7].

2.3 XGBoost

XGBoost is an acronym for eXtreme Gradient Boosting [6]. The algorithm is an implementation of a gradient boosting machine [8]. Gradient boosting machines are based on an ensemble of decision trees, as discussed in section 2.1, where multiple weak learner trees are used in combination as a collective to give better predictions than individual trees can do [7]. In comparison with older gradient boosting algorithms, XGBoost has superior regularization and better handling of missing values, as well as much improved efficiency [6].

2.4 Boosting

The concept of boosting is to use an ensemble of weak learners to iteratively improve on the results of the previous model. The *AdaBoost* algorithm, as discussed in [7], consists of a sequence of weak classifiers. In each iteration, the best classifier is identified based on the current sample weights. Any classification errors in the current iteration receive more weight in the next iteration, while the inverse is true for correctly classified samples. Each iteration focuses on a different aspect of the data, such that regions of data that are difficult to classify are treated separately. In the final stage, an ensemble is created from the combined overall sequence of classifiers. This ensemble is likely to have a better prediction performance than any individual classifier [7].

3 The activity recognition data set

The data set used in this project [2] and was collected by using a Samsung Galaxy S2 attached to the waist of 30 volunteer test subjects. Data was recorded from accelerometers and gyroscopes as the subjects performed six typical daily activities. The activities were classified as horizontal walking, walking upstairs, walking downstairs, sitting, standing and laying.

The raw data consist of six channels of data measuring the body acceleration and angular motion from the gyroscope in three spatial directions. A sampling rate of 50Hz was used. These sensor signals where pre-processed by applying a selection of methods, described in [2]. Notably, the acceleration signal is separated in its gravitational and body motion component. The measurements were filtered using a sliding window filter. Finally, a vector of features was calculated giving a total of 561 predictors used in this project.

3.1 Independent training and test set

The data set was originally split into training and test sets by randomly selecting 30% of the human test subjects for the test set [2], such that the subject pools for each dataset are disjunct. This ensures that the test data is independent of the training data. The training and test set contains 7352 and 2947 samples, respectively. It is reasonable to expect significant variation between how different subjects perform the same activity, e.g. different walking patterns or walking gaits. Therefore, which subject is performing an activity will also influence the data in addition to the activity itself. Further, since the data is recorded as time series during an activity, multiple samples from the same person doing the same activity is included in the full data set. Hence any form of random selection of samples, i.e. bootstrapping or cross-validation (CV), will not produce independent data sets. It is likely that any such subsampling will cause the same subject and activity combination to end up in multiple sub-sets. This effect is apparent later when tuning hyper parameters using CV on the training set. As the results will show, the accuracy estimates generated by CV are overly optimistic when compared to accuracy estimates performed on a proper and independent test set. Regardless, the CV accuracy results are assumed to be usable for parameter tuning, but should not be taken as estimates of the methods ability to predict independent data.

4 Analysis and results

4.1 Correlation between predictors

With such a large number of predictors, which all are derived from the same six sensor channels, it is natural to expect a high degree of correlation between variables. For the decision tree methods used in the following, such correlation can present some difficulty, since it makes the choice of optimal predictor on which to split somewhat arbitrary [7].

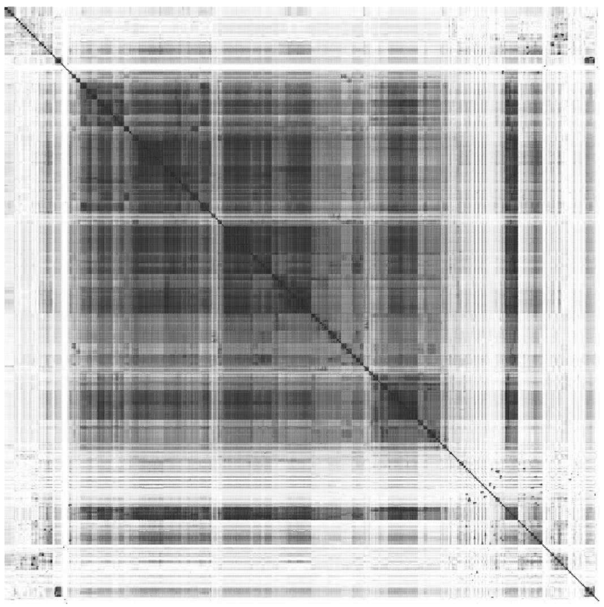

Figure 1 - This plot shows the correlation matrix for all 561 predictors. Such a large matrix is difficult to analyze manually. The R package *corrplot* is used to organize the predictors such that predictors with high correlation are plotted together in groups. Dark color signifies high correlation while low correlation is plotted in white. While detailed analysis of such a plot is difficult when the number of predictors is large, it is useful to investigate the structure of the correlation matrix.

If multiple predictors contain essentially the same information, i.e. the same relevance to the classification process, the tree algorithms will have degraded prediction performance due to this irrelevancy in the choice of which predictor gives the optimum split at each tree node [7]. To highlight the problem, a correlation analysis of the data is performed using the R function *cor*. This calculates inter-predictor correlations in a data matrix, and the *corrplot* method from the library *corrplot* can be used to plot the results.

The correlation matrix plot in Figure 1 shows that the correlation between variables, or predictors, is extensive. The plot shows essentially three groups of correlated predictors and it may be concluded that there are a large number of predictors giving essentially redundant information.

4.2 Principal Component Analysis (PCA)

A natural next step is to use PCA to inspect the data further [11]. The software package Unscrambler X by CAMO is used to perform PCA. This decomposes the data into principal components, in the form of scores and loadings [12]. These plots can be interpreted to gain insight into the important latent structures in the data. A principal component (PC) can be thought of as a hidden variable, or a latent structure in the data. It consists of a direction in the predictor space which maximizes the variance of the samples. Each PC is normal to all the other PC's, such that there is no correlation between them. The score-plot can be used to identify grouping of samples. Loading-plots show the effect of all the variables on the PC's.

Prior to computing PCA, the data is pre-processed with centering and scaling. The centering transformation subtracts the average value from each predictor, such that the new data is zero centered. The scaling transformation divides each predictor by its standard deviation. This has the effect of transforming the data such that all predictors have a standard deviation of one, thus influencing the computation of PC's equally.

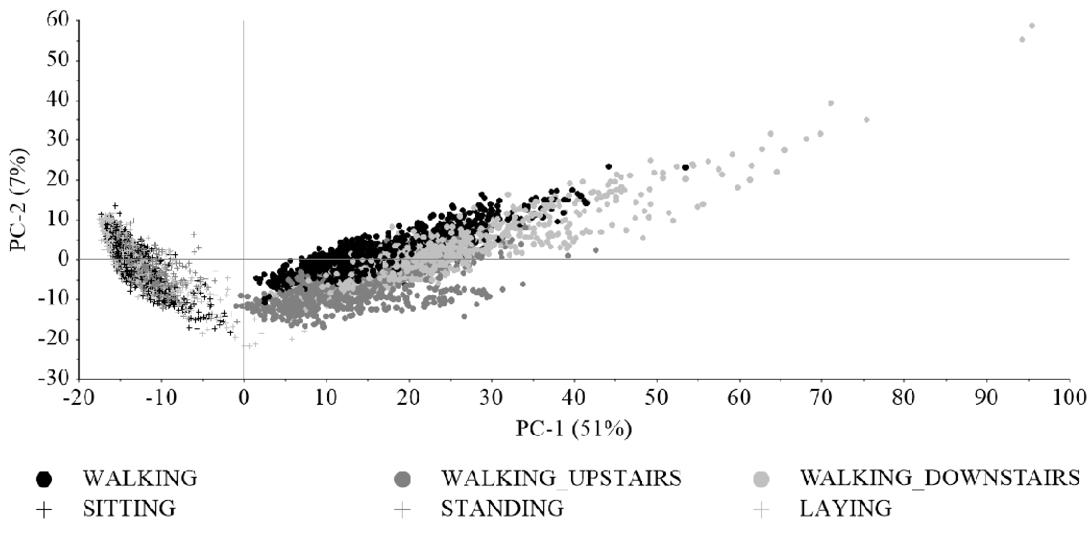

Figure 2 - Scoreplot from PCA, including color legend for each of the six activities. The plot shows that by using PC-1 and PC-2 it is possible to separate static and dynamic activities.

In Figure 2, as shown by the legend, each of the six response classes is highlighted by gray-tone and marker shape. Dynamic behaviors is marked with dots while stationary behaviors are marked with cross. The horizontal axis shows PC-1, the most dominant PC, which accounts for 51% of the total variation in the data set. By inspection of this plot, it is clear that there are two groups of samples. The three classes that contain samples from subjects that are walking are clearly separated from the other three classes where the subject is stationary. Furthermore, in the PC-2 direction, containing 7% of the total variation, the three walking activities are further separated but with significant overlap. This analysis shows that it is reasonable to continue with classification, since there is clearly information in the predictors that correlate to the activity class, i.e. there is some structure in the data that can be used by algorithms to build classification models.

4.3 Random Forest

The Random Forest method is the first method used to classify the data. However, due to the high number of predictors and samples in the aforementioned training data, several deviations from suggested "best practice" in [7] in terms of tuning parameters were done here. This tuning of the parameters will be discussed in a subsequent sub-section. In ref [7], Kuhn *et.al*. suggest tuning m_{try} from 2 to P, where P is the number of predictors, in five evenly spaced values. The initial test here is rather performed with values up to 100 randomly selected predictors at each step. The initial test is done with single 5-fold cross validation (not repeating).

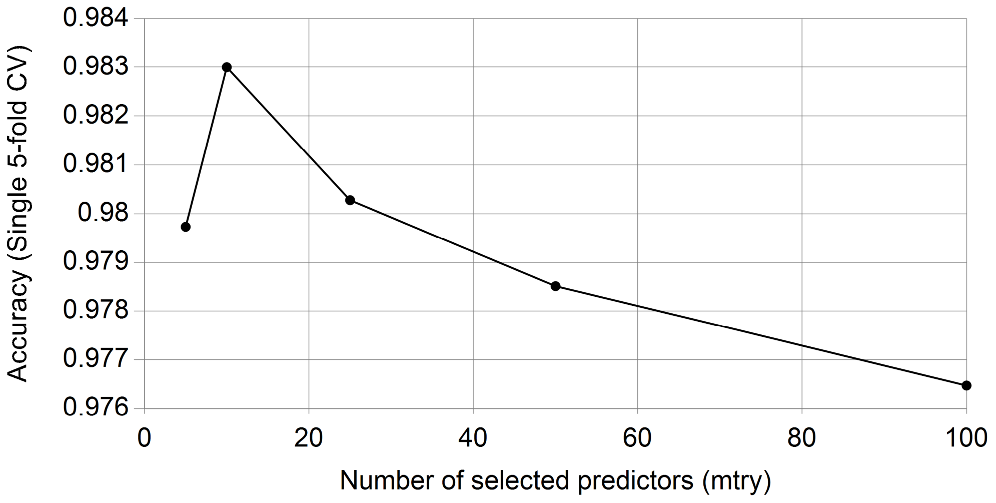

Figure 3 - First try at Random Forest, showing prediction accuracy vs m_{try}.

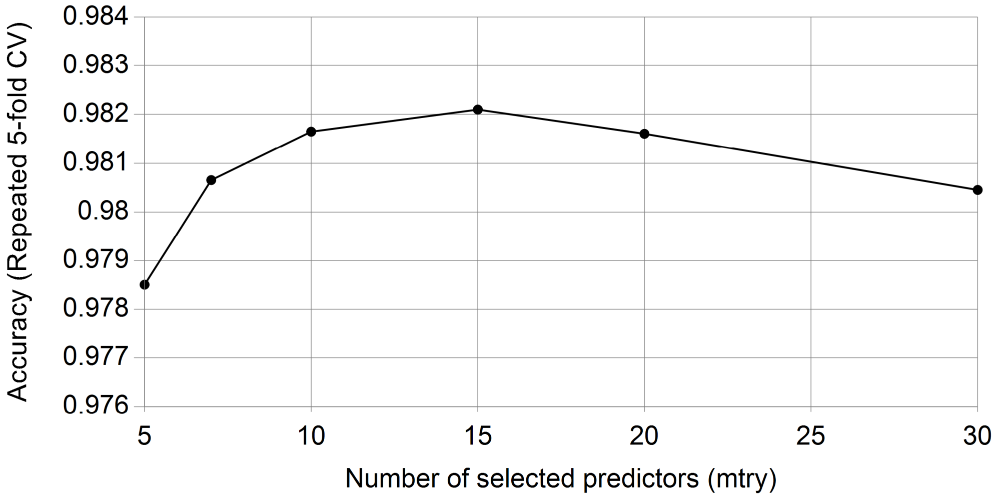

Figure 4 -Tuning results from the Random Forest method. The plot shows prediction accuracy using 5-fold CV with 25 repeats against the tuning parameter mtry.

As shown in the initial test of RF in Figure 3, the performance is not drastically affected by the number of randomly selected variables m_{try}, however the accuracy deteriorates if m_{try} is above 15 predictors. This value corresponds approximately to Breimans suggestion [10] of setting m_{try} to the square root of P (~23). The lack of improvement with higher m_{try}, i.e. including more data, could be related to the high degree of correlation between the predictors in this data set. However, according to [7] , this lack of sensitivity to m_{try} is a commonly seen result.

A more extensive test of RF, using a higher number of CV folds and repeats, is shown in Figure 4. The optimal number of predictors in each iteration appears to be 15. The difference in accuracy between the tested choices of tuning parameter is not large. It is important to note again, as discussed in section 3.1, that the CV based accuracy estimates are only intended for parameter tuning, and must not be taken as estimates of method accuracy.

Kuhn et.al. recommends at least 1000 trees to be used in RF, but that would take some ~10 days to compute with a reasonable amount of cross validation (CV), say 25 repeats of 5-fold CV, for this data se, hence only 100 trees are used here.

4.4 XGBoost

The next method is XGBoost. The R implementation of this method is taken from [13]. This implementation is efficiently parallelized which provides fast computation. The gradual stepping up in complexity is therefore not necessary. Instead, a full run with a default tuning grid is executed immediately.

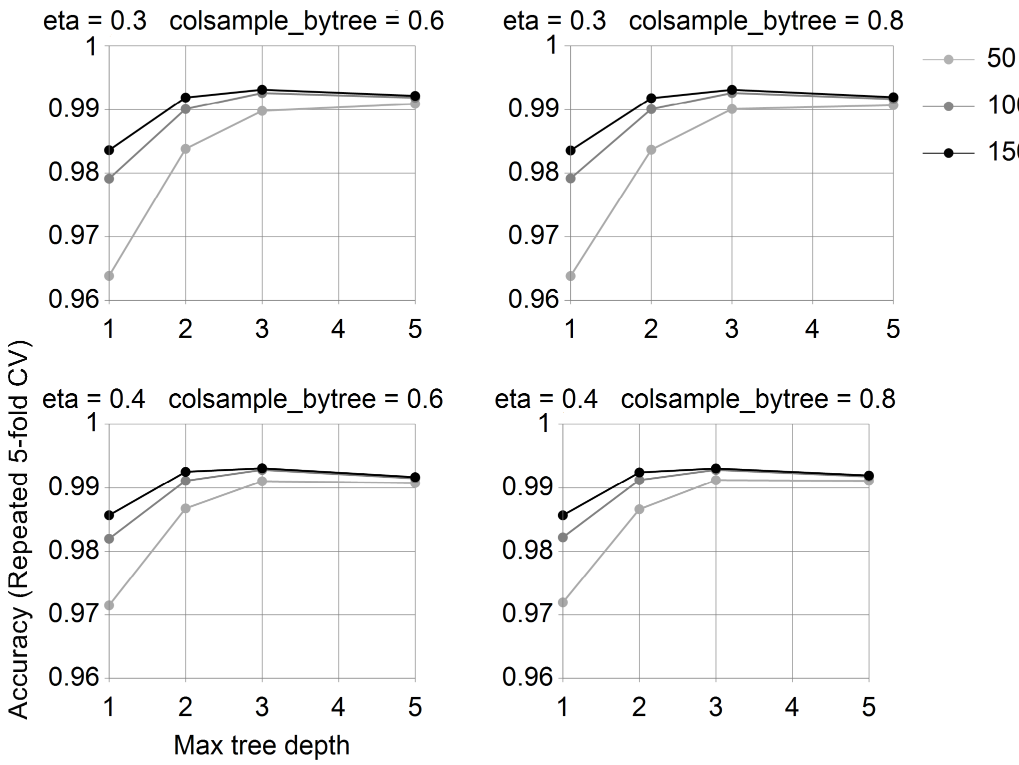

Figure 5 - Shows tuning results for XGBoost. For this method there are 4 tuning parameters. The figure consist of four sub-plots, each representing particular settings for eta and colsample parameters. The sub-plots shows the maximum allowed tree-depth on the absica and the prediction accuracy calculated from 5-fold CV repeated 25 times on the ordinate axis. Each sub-plot have three graphs, one for each of the three tested settings of boosting iterations (50, 100 and 150).

Despite using a higher number of tuning parameters, resulting in several times more runs then previous algorithms, XGBoost finishes in ~4 hours. A higher number of boost iterations is also used here, from 50 to 150. The efficiency of this implementation is further evidenced by the CPU load holding steady at 90% throughout the computation.

As shown by Figure 5, the tuning grid includes variations over the typical training parameters for XGboost. The *eta* and *colsample* parameters have little effect on the model accuracy. The *eta* setting controls the learning rate of the algorithm by scaling the contribution of each new tree. The *colsample_bytree* setting determines the ratio of predictors that are used for each new generated tree. For a tree-depth of 1 there is some observable improvement by increasing eta from 0.3 to 0,4 but for higher tree-depths there is no difference for either of these two parameters.

Max Tree Depth and boost iterations are more important. In all cases, the accuracy is in the range 95-99%. These results are found using 5-fold cross validation repeated 25 times. As discussed in section 3.1, CV does not give independent sub-sets of the data used here. Hence the accuracy estimates can only be used for parameter tuning and does not represent accurate predictions of the method accuracy.

4.5 C5.0

C5.0 results are only used as a reference method for Random Forest and XGBoost. As such, detailed discussion of the tuning for C5.0 is not included here. However, the method was tuned using a similar approach to that presented for Random Forest and XGBoost. The results of tuning hyper parameters show that 30 boosting iterations is a good choice. Further, there is no significant improvement by application of winnowing.

4.6 Method accuracy estimated on independent test set

The cross-validation (CV) results produce model prediction accuracies close to 100%. As discussed in section 3.1, the time-series data in this set is not well suited for CV; hence these estimates are not good predictions of model accuracy. When drawing training samples randomly using CV it is highly likely that the same subject performing the same activity will end up in several folds. Consequently, the folds of CV are not truly independent. In order to accurately assess the methods ability to predict new data, an independent test set is needed.

For each method in the subsequent sections, a confusion matrix showing the classification results with both estimated and reference class is given. Additionally the sensitivity and specificity for each class is stated in a separate table. Discussion of the results is deferred to section 5. For all confusion matrices the predicted class is shown in the row direction while the reference class is shown in the column direction

4.7 Random Forest on test set

The best hyper-parameter tuning for Random Forest was previously found to be 30 boosting trials without winnowing.

Table 1 - Confusion matrix Random Forest on the test set

	Walking	WalkUp	WalkDown	Sitting	Standing	Laying
Walking	481	30	20	0	0	0
WalkUp	4	435	45	0	0	0
WalkDown	11	6	355	0	0	0
Sitting	0	0	0	446	26	0
Standing	0	0	0	45	506	0
Laying	0	0	0	0	0	537

Table 2 - Sensitivity and specificity for Random Forest on the test set

	Walking	WalkUp	WalkDown	Sitting	Standing	Laying
Sensitivity	97.0	92.4	84.5	90.8	95.1	100.0
Specificity	98.0	98.0	99.3	98.9	98.1	100.0

The classification accuracy on the test set is found to be 93.7% for the Random Forest model. The confusion matrix is shown in Table 1 and the sensitivity and specificity is shown in Table 2.

4.8 XGBoost on test set

The best hyper-parameter tuning for XGBoost was previously found to be 150 boosting iterations with a max tree depth of 3.

Table 3 - Confusion matrix XGBoost on the test set

	Walking	WalkUp	WalkDown	Sitting	Standing	Laying
Walking	490	31	6	0	0	0
WalkUp	6	433	23	2	0	0
WalkDown	0	6	391	0	0	0
Sitting	0	1	0	431	26	0
Standing	0	0	0	58	506	0
Laying	0	0	0	0	0	537

Table 4 - Sensitivity and specificity for XGBoost on the test set

	Walking	WalkUp	WalkDown	Sitting	Standing	Laying
Sensitivity	98.8	91.9	93.1	87.8	95.1	100.0
Specificity	98.5	98.8	99.8	98.9	97.6	100.0

The classification accuracy on test set is found to be 94.6% for the XGBoost model, which is a good result. The confusion matrix is shown in Table 3 and the sensitivity and specificity is shown in Table 4.

4.9 Comparing methods

The sensitivity scores presented for each method in the previous sub-chapters show how well each of the methods is able to classify the examples in the test set. It is now of interest to compare them in tabulated form, including the results from the original paper. The same training and test sets are used here [3].

Table 5 - Comparing accuracy in percent for different methods and activity classes

	Walking	Walk Up	Walk Down	Sitting	Standing	Laying	Overall
C5.0 trees (boosted)	93.8	91.5	88.3	81.5	91.0	100.0	91.2
C5.0 rules (boosted)	97.0	96.4	91.9	87.0	89.7	100.0	93.7
RF (highly tuned)	97.0	92.4	84.5	90.8	95.1	100.0	93.7
XGBoost (highly tuned)	98.8	91.9	93.1	87.8	95.1	100.0	**94.6**
SVM (From [3])	95.6	69.8	83.2	96.4	93.0	100.0	90.2

From Table 5 we may conclude that the results are similar to those of [3]. The overall accuracy is computed as a weighted sum over all test examples. As mentioned previously, the XGBoost method performs slightly better then Random Forest and rule-based C5.0.

An interesting observation from e.g. Table 3 is that there are three disjunct groups of activities where there are no miss-classifications between these groups. The first group consists of the three walking activities, which are separate from the other three stationary activities. This is expected based on the results from PCA, where the first PC clearly shows that these two types of activities, i.e. walking/dynamic vs stationary, can be separated by only one PC. Further, the activity laying also has no sample misclassifications. This can likely be traced back to the angle of gravitation predictor, since the laying activity is distinctly different in this respect compared to the other five classes of activities

5 Discussion and conclusion

The process of identifying human activity based on simple accelerometer and gyroscope data has been the goal of the present research. Since the data is difficult to analyze using physical models, machine learning methods [7] were applied.

In this investigation we used techniques based on C5.0, Random Forest and XGBoost. These methods were used to classify activities such as walking, walking downstairs, walking upstairs, sitting, standing and laying

To remedy the challenges of cross-validation (CV) discussed in section 3.1, a full test set was used to estimate the methods accuracy. The data set contains a random selection of 30% of the test subjects in a separate data set. This is used for estimating the prediction accuracy of all four applied methods.

From the correlation plot in Figure 1 it appears that there are three main groups of variables. This matches well with the confusion matrices from both methods, which show that the laying activity is distinct from the other five to such a degree that it can be ~100% classified with only one split of the data, or equivalently one rule for the rule-based models. Further, the dynamic activities, i.e. walking, are different from the stationary activities of standing, sitting and laying. This is also shown by application of PCA in Figure 2 Here the first principal component is used to separate dynamic and stationary activities.

The best results on the independent test set are found with XGBoost (*nround* = 150, m*ax_depth* = 3). This method shows a prediction accuracy of ~94.6%. Random Forest (*mtry* = 15) and C5.0 rules (*trials* = 30) show equal prediction accuracy of ~93.6%, while C5.0 tree based (trials = 30) is behind the other three at 91.2%. It is somewhat unexpected that C5.0 with rules outperforms C5.0 with trees by 3pp. This could be related to improved generalization in the rule-based model, since these accuracy estimates are taken using an independent test set.

The sensitivity ratings for all classes and models show some class dependent variation between the four methods. For example, Random Forest has a slightly better sensitivity to the sitting activity compared to XGBoost, even though the overall accuracy for XGBoost is better.

The results achieved here with XGBoost are significantly better than the SVM results published in [3]. The main increase in accuracy comes from the tree-ensemble methods used here being able to separate the Downstairs class better then the SVM. The improvement in classification of the dynamic activities is significant. For the stationary activities, the methods are similar. The XGBoost algorithm performs the best overall.

For all methods, we find three groups of activity classes in the confusion matrices. The dynamic activities of walking, walking downstairs and walking upstairs form one group. Stationary vertical activities, i.e. standing and sitting, form a second group, while laying constitutes the third group. Within each group there is a near 100% classification success rate, indicating that these three groups of activities are distinctly separable by all the applied methods. The difficulty is separating activities within each group. In particular, based on specificity and sensitivity values, it is difficult to separate standing and sitting. However, also these activities show a sensitivity of around 90% for the XGBoost method.

The results of this project show promise in classification of human activities based on inertial sensor data. The classification success rate is 94.6% for the best method, which is a large improvement over published SVM results [3]. For such a strong claim to be justified, the evidence of this success rate should be substantial. Based on the test set validation using different test subjects to record training and test data, the estimated prediction accuracy is considered to generalize, i.e. it accurately represents the prediction of new samples.

6 Suggestions on Future Studies

The ideal scenario for most applications of machine learning is to use a three-way split of data into training, validation and test sets. In this way the data set from separate subjects could be used for all three phases of model development, training of model, tuning of parameters, and testing of the final results. It would be of interest to verify that the model tuning results presented here are indeed optimal also when an independent set of samples is used to measure the accuracy of each combination of hyper parameter values.

Another interesting possibility is improving the separation of body and gravitational acceleration vectors. This is achieved with a Butterworth filter in [2]. It is reasonable to assume that the performance could be improved using e.g. a sensor fusion model and a Kalman filter [14]. A reduced form of this method is often called a complimentary filter. The method integrates the gyroscope signal and uses a high-pass filter to eliminate integral drift. A low-pass filter with an identical cut-off frequency to the high-pass filter is applied to the accelerometer, before both filtered outputs are summed together. This method could possibly retain more dynamic information from the raw data, compared with the Butterworth filter applied in the original article [5].

7 Acknowledgments

The authors would like to thank Clark S. Lindsey for his help in reading and commenting the manuscript. We would also like to thank Jorge L. Reyes-Ortiz, Davide Anguita, Alessandro Ghio, Luca Oneto and Xavier Parra for making the complete dataset available as public domain [2].

8 Bibliography

[1] V. G. Sanchez and C. F. Pfeiffer, "Legal Aspects on Smart House Welfare Technology for Older People in Norway," *Inteligent Enviroments 2016*, 2016.

[2] J. L. Reyes-Ortiz, D. Anguita, A. Ghio, L. Oneto, and X. Parra. (2012). *Human Activity Recognition Using Smartphones Data Set*. Available: http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

[3] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine," in *Ambient Assisted Living and Home Care: 4th International Workshop, IWAAL 2012, Vitoria-Gasteiz, Spain, December 3-5, 2012. Proceedings*, J. Bravo, R. Hervás, and M. Rodríguez, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 216-223.

[4] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, "Energy Efficient Smartphone-Based Activity Recognition using Fixed-Point Arithmetic," *Journal of Universal Computer Science*, vol. 19, 2013.

[5] J. L. Reyes-Ortiz, A. Ghio, X. Parra-Llanas, D. Anguita, J. Cabestany, and A. Català, "Human Activity and Motion Disorder Recognition: Towards Smarter Interactive Cognitive Environments," presented at the 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013, 2013.

[6] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *KDD 2016*, 2016.

[7] M. Kuhn and K. Johnson, *Applied Predictive Modeling*: Springer, 2013.

[8] J. H. Friedman, "Greedy Function Approximation: A Gradient Boosting Machine," *The Annals of Statistics*, vol. 29, pp. 1189-1232, 2001.

[9] (04.02). *Introduction to Boosted Trees*. Available: http://xgboost.readthedocs.io/en/latest/model.html

[10] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, pp. 5-32, 2001.

[11] K. Esbensen, *Multivariate analysis in practice*. Oslo: CAMO, 1998.

[12] H. Martens and T. Næs, *Multivariate calibration*. Chichester: Wiley, 1989.

[13] T. Chen, T. He, M. Benesty, V. Khotilovich, and Y. Tang. *Extreme Gradient Boosting*. Available: https://cran.r-project.org/package=xgboost

[14] H. J. Lee and S. Jung, "Gyro sensor drift compensation by Kalman filter to control a mobile inverted pendulum robot system," in *2009 IEEE International Conference on Industrial Technology*, 2009, pp. 1-6.