

Posture Classification Based on a Spine Shape Monitoring System

Icxa Khandelwal[1(✉)], Katharina Stollenwerk[2], Björn Krüger[3], and Andreas Weber[1]

[1] University of Bonn, Bonn, Germany
icxakhandelwal04@gmail.com
[2] Hochschule Bonn-Rhein Sieg, Sankt Augustin, Germany
[3] Gokhale Method Institute, Stanford, CA, USA

Abstract. Lower back pain is one of the leading causes for musculoskeletal disability throughout the world. A large percentage of the population suffers from lower back pain at some point in their life. One non-invasive approach to reduce back pain is postural modification which can be learned through training. In this context, wearables are becoming more and more prominent since they are capable of providing feedback about the user's posture in real-time. Optimal, healthy posture depends on the position (sitting, standing, hinging) the user is in. Meaningful feedback needs to adapt to the current position and, in the best case, identify the position automatically to minimize necessary interactions from the user. In this work, we present results of classifying the positions of users based on the readings of the *Gokhale SpineTracker™* device. We computed various features and evaluated the performance of K-Nearest Neighbors, Extra Trees, Artificial Neural Networks and AdaBoost for global inter-subject classification as well as for personalized subject-specific classification.

1 Introduction

Before diving into the matter, we need to fix some terminology. While generally used interchangeably in everyday language, throughout this paper we will differentiate between *position* and *posture*. *Position* will refer to the (static) state a human body is in, e.g., sit, stand, bend, hinge. *Posture* describes how a position is realized by a person. Thus, a position resembles a passive state description whereas posture becomes a multi-factor dynamic process. Factors therein include bone or, more general, skeletal alignment as well as muscle activation.

According to the Global Burden of Diseases, Injuries, and Risk Factors Study 2016 [1], lower back pain is one of the leading causes for years lived with disability throughout the world. It was reported that two major causes of musculoskeletal disability are lower back pain and neck pain. Due to the sedentary lifestyle and long sitting hours at work, a large number of people are at a high risk of developing spine related issues. The low back pain fact sheet [2] explains that people who are not physically fit have weak back and abdominal muscles.

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11624, pp. 494–509, 2019.
https://doi.org/10.1007/978-3-030-24311-1_36



Fig. 1. Placement of the SpineTracker sensors on a subject's back. Posture training with real-time feedback of the spine shape. Screenshots of the iPhone app: Spine curve and Spine simulation.

Consequently, these muscles cannot properly support the spine and thereby cause pain in the long run. If, in addition, such individuals are obese, more stress is put on the muscles, bones, and discs at the back which also leads to pain.

Gokhale Method Enterprise Inc.[1] is a Silicon Valley-based company specialized in posture training. According to this company's philosophy, the key to a healthy spine is healthy lifestyle and correct posture. Hence it provides various training classes to interested customers so that they can improve their posture. Gokhale Method has developed several posture supportive products, one of them being the *Gokhale SpineTracker™* to support posture training in classes. This device comprises of five wearable sensors equipped with three-axis accelerometers. It is attached along the spine of the user to provide a reasonable approximation of user's spine shape. The data obtained through this device can be visualized in real-time via an iPhone application [3]. See Fig. 1 for some examples.

Since healthy spine shape strongly depends on the current position, an automatic detection of the position would be highly beneficial. In this work, we use data of the *Gokhale SpineTracker™* to automatically detect the position the user assumed. We have done so by using supervised machine learning techniques for the classification of a user's posture into different classes (positions), i.e., sitting, standing, and hinging. These positions are especially challenging to be classified from accelerometer readings as they are mostly static. Thus, there are only slight changes in the sensor readings over time. In particular, this makes the distinction between sitting and standing very hard. This problem is further increased by the variability of postures between subjects.

The remainder of this paper is organized as follows: We give an overview on related work in Sect. 2. Section 3 briefly describes the datasets (Sect. 3.1) this work is based on and covers feature computation (Sect. 3.2). An overview on used classification techniques is given in Sect. 4. The different classification experiments performed in this work are described in Sect. 5. We present the classification results in Sect. 6 and conclude the paper in Sect. 7.

[1] https://gokhalemethod.com/.

2 Related Work

Motion Capturing Techniques. Capturing and recording of human motion data has become a standard technique in may domains, such as computer animation, sport sciences [4], biology [5], and rehabilitation [6]. Optical motion capture with passive markers and a large array of cameras has become the gold standard of capturing motions due to its high temporal and spatial resolution and accuracy [7]. However, optical motion capture is limited in the capture volume and its general usability. The user has to wear a motion capture suit and undergo a complex calibration process which is not suited for everyone. Thus, alternative techniques based on inertial sensors, especially with few accelerometers only, were developed to capture full body motions [8–11]. While full body motion capture based on few sensors gives a good overview of the general motion, it is not yet possible to capture all body parts in high spatial resolution. Specialized systems were developed to more accurately capture specific body parts, such as faces [12,13], arms [14], legs [15], hands [16–18], and the spine [19,20].

Supervised machine learning techniques have proven to be especially useful in the field of computer vision-based human motion capture. Examples hereof include capturing of motions from video [21,22] and retrieval of single poses from one image [23,24]. Machine learning has also been successfully applied for reconstruction of motions from other sensor modalities [8–10]. An application beyond the capturing of motions is the classification of motions. Here, identifying certain actions [25,26] was done as well as the detection of biometric parameters [27].

Spine Shape Reconstruction and Motion Classification. A vast variety of systems and technologies have evolved around the capture the shape of the spine for posture monitoring. Such technologies range from ribbon-shaped fibre-optic sensors [19], over strips of strain-gauge elements [28], accelerometers [20,29] to various inertial sensor-based systems [30–33]. Many of the aforementioned systems use posture or activity classification as field of application.

StraightenUp+ [33] is a vest with three inertial sensors vertically mounted to the vest's back. Using the sensor data of 30 participants, the authors use decision trees in order to identify physical activities such as sitting and walking from a fixed sequence of eight activities. Wong and Wong [30] developed a smart garment with three inertial sensors for posture training. Their system defines posture change as change in inclination between pairs of neighboring sensors. In a small posture-feedback study, this information is used as an indicator for the straightness of the spine and hence an indirectly indicator for, e.g., slouching. In a larger study (429 participants) Consmüller et al. [28] used the Epidonis SPINE system to record five repetitions of a choreographed sequence of back movements consisting of rotations in the sagittal, frontal, and transverse plane. Using linear discriminant analysis they successfully classify the resulting back motions. Fathi and Curran [32] place three inertial sensors on cervical, thoracic, and lower lumbar spine to record and classify two positions (hunch, slouch) using template-string matching. Distributing five 3D accelerometers equidistantly over the spine, Jeyhani et al. [29] differentiate sitting positions (straight, hunch, arch) based on trends in the per-sensor mean inclination.

There are generally few works for posture classification that use real-world data and consist of a large amount of data from a wide range of users. This is especially true for classification of spine shapes (postures) using wearables. This paper contributes to filling this gap by providing an evaluation of different classification techniques for both inter-subject and subject-specific posture classification of postures in three different positions that are meaningful for posture training in a database of almost 4,200 spine movements from over 100 subjects.

3 Data Processing

This section provides a general overview of the data and data processing steps used in this work. It includes details on different datasets and a description of features derived from the raw data.

3.1 Datasets

The dataset is obtained from the sensors via the iPhone application for *Gokhale SpineTracker™*. The sensors' data are stored in a binary file, which is uploaded to one central server. Per default, data are captured with a sampling frequency f of 50 Hz. Each file contains the following relevant information:

Position refers to the position the user assumed. In this work, we focus on *standing*, *sitting*, and *hinging* as these are part of the standard posture training. Other positions are ignored also because we do not have sufficient samples for learning approaches.

Event describes the event occurring at the end of each motion sequence. User interactions trigger events in the iPhone application (closed application, connection aborted, or switched posture for instance). In this work, we are interested only in sequences that are ended by the event *snapshot*. This event signifies that the user took a snapshot of his pose and confirmed the position he has taken the snapshot from. Therefore, the position label is reliable in this case. In other events the user might have switched to another pose, without changing the position in the application. Thus, there would be many mislabeled data for other events, which is not suitable for supervised learning approaches.

Readings contains a table with timestamps and the raw sensor data.

From the data uploaded by the application, a set of data was extracted for this work. This raw dataset ($\mathcal{DB}_{\text{raw}}$) contained 15,873 files, belonging to 150 subjects. Preprocessing, included removal of files corresponding to the wrong event type, removal of corrupt files, and removal of files with less than one second of data. The complete dataset ($\mathcal{DB}_{\text{full}}$) we continued to work with contained 4,178 files from 106 subjects, finally. For some experiments we separated a partial dataset ($\mathcal{DB}_{\text{partial}}$) consisting of 1,457 files covering 37 subjects.

3.2 Feature Computation

We computed various types of features based on the sensors' accelerometer data. Each of the five sensors captures acceleration values in three axes resulting in a 15-dimensional multivariate time series of raw accelerations. Features were computed on windows of fixed temporal length s ($s \in [1, 10]$ s), varying the time before a snapshot event occurred. Based on these fixed-length segments, we compute different feature types as listed below:

Raw features Here, the original, raw data obtained from the sensors are taken as feature. This results in a $(15 \cdot f \cdot s)$-dimensional feature vector.

Statistical features: For each of the 15 time series, we computed eight basic statistical measures: length, mean, standard deviation, minimum value, maximum value, median, 25% quantile, and 75% quantile. This results in a 120-dimensional feature vector.

Fourier features The absolute Fourier transform of the data was considered for this feature type. For each of the 15 time series, their discrete Fourier transform was computed using the `numpy.fft.fft()` function of Python's scipy library [34]. The result is $(15 \cdot f \cdot s)$-dimensional feature vector.

Angular features This feature type stores the *forward tilt* τ_{acc} of each sensor. For static poses, it can be assumed that the sensor is not displaced from its position. As a consequence, that sensor will only measure acceleration due to gravity pointing downwards, and we can compute its *forward tilt* as

$$\tau_{\text{acc}} = \arctan2\,(a_z, -a_y), \tag{1}$$

where a_z and a_y denote the acceleration of the sensor measured in its local z-axis and y-axis respectively [20]. For each frame, we obtain five forward tilts, one from each sensor. Additionally, the difference of forward tilt between two consecutive sensors was computed (four values for each frame). Hence the dimensionality of this feature set is $((5 + 4) \cdot f \cdot s)$.

3.3 Principal Component Analysis

Principal Component Analysis (PCA) is one of the most common linear dimensionality reduction techniques. In a first step, a linear mapping of the feature space is computed, where dimensions are sorted by variance in the data. Dimensions in this space are called Principal Components (PCs). In a second step, one can truncate PCs representing small variance in the data. This allows for the reducing the dimensionality, while maintaining control over the degree of data loss in the compressed space.

4 Supervised Machine Learning Techniques

We provide a short introduction to the supervised machine learning (SML) techniques used for posture classification in this section. All of the described techniques have a specific set of parameters. The concrete parameter values we used

are listed and described in Table 1. Machine learning techniques generally require the dataset to be split into two independent datasets, one for training the classifier (training dataset) and one for assessing its performance (validation dataset). After training, the classifier can be used for making predictions on the unknown dataset (test dataset).

Table 1. List of classifiers, parameters, their short description, and values used in this work for classification. The parameter n specifies the number of samples in the training set. A detailed description of the parameters can be found at scikit-learn [34].

Classifier (model)	Parameter	Description (values used)
KNN	k	Number of nearest neighbors ($k = n/d$, $d \in \{10, 20, \ldots, 100\}$)
	w	Weighing of neighbors ($w \in$ {uniform (U), distance-based (D)})
Extra-Trees	e	Number of decision trees ($e \in \{10, 100, 250, 500\}$)
	f_{sq}	Function to measure the quality of a split ($f_{\text{sq}} \in$ {Gini impurity (G), entropy (E)})
ANN	h_i	Number of neurons per hidden layer
	f_{act}	Hidden layer activation function ($f_{\text{act}} \in$ {identity (i), logistic sigmoid (l), tanh (t), rectified linear unit (r)})
	f_{opt}	Weight optimization function ($f_{\text{opt}} \in$ { family of quasi-Newton methods (l), stochastic gradient descent (s), specific stochastic gradient-based (a)})
	l	Type of learning rate ($l \in$ {const. (c), invscaling (i), adaptive (a)})
AdaBoost	e	Number of decision trees ($e \in \{10, 100, 250, 500\}$)
	c	Classifier used (all possible combination of Extra-Trees)

4.1 *K*-Nearest Neighbor Models

K-Nearest Neighbor (KNN) [35] classifiers store features of the training dataset as vectors. For classifying a test instance, the distance between that test element and all elements in the training dataset is computed. Distance metric used most commonly for this purpose is the Euclidean distance. For a pre-defined number K of closest elements of the training set, a majority vote is performed. The class belonging to most of the K neighbors is assigned to the test sample. It has been observed that this classifier is successful for datasets in which the decision boundary is very irregular [36].

Bhatia et al. [37] describe the strengths and weaknesses of KNN. Advantages of this classifier are that it trains very fast, is robust to noisy training data and is effective on large training datasets. One of its biggest limitation is that the optimal value of K needs to be determined. Additionally, it is not clear which distance metrics and attributes shall be used to obtain best possible results. Weighted KNN [37] technique can be used to overcome the inherent limitation of KNN to assign equal weights to K nearest neighbors. In this technique, weights are assigned to the neighbors based on their distance from the test instance. In

this work, KNN models are created using the function `KNeighborsClassifier()` of scikit-learn library of Python [34] and the weights parameter is varied as either uniform or distance based weights.

4.2 Extremely Randomized Trees (Extra-Trees) Models

In a decision tree [38], leaf nodes represent classes and internal nodes represent different tests on a set of features. The branches depict the flow in the tree based on logical conjunctions of nodes considered until now. The aim is to develop a model which can predict the class of a test dataset element based on a selected set of features. It employs top-down divide and conquer by considering the information entropy of different features. The complete process [35] consists of three steps: 1. Determine feature and threshold to partition the training set at each internal node. 2. At each node check whether to continue splitting or to make it a leaf node. 3. Assign class labels to leaf nodes with minimum estimated error rate.

According to Ho [39], one of the biggest advantages of decision trees is that they are easily to interpret. They perform well on large datasets, but might learn highly irregular patterns from the data if the trees are very deep. Thus the resulting model will highly fit the training data but may perform poorly on test data (overfitting). To overcome this limitation, meta estimators of decision trees such as Extra-Trees (ET) [40] are used. Extra-Trees combine multiple decision trees which are trained on different sub-samples of the dataset. The overall performance is determined by calculating the average of performance of all decision trees. In this work, we used the extra trees model implemented in `ExtraTreesClassifier()` function of scikit-learn library of Python [34].

4.3 Artificial Neural Network Models

Artificial Neural Network models (ANN) simulate the functioning of neurons in the brain. The most common ANN model used is a multi-layer perceptron with backpropagation [38]. Its architecture [35] consists of three types of layers: 1. The *input layer* feeds the feature vector of the training set to the first hidden layer. 2. One or more *hidden layers* weight all prior outputs differently and add a bias term. 3. The *output layer* processes the output of the final hidden layer. It contains a unit representing each class.

ANN [41] are capable of detecting complex non-linear relationships between classes and the features associated with the training data. Using ANN it is possible to detect all possible interactions between the features. But these networks exhibit black box behavior, i.e., it is not possible to identify the exact causal relationship between the variables which lead to a particular outcome. In this work, ANN models are created using `MLPClassifier()` function of the scikit-learn library [34].

4.4 Adaptive Boosting Models

Adaptive boosting (AdaBoost) is a meta estimator. It constructs a strong classifier using weak classifiers which can be any other supervised machine learning algorithm with low accuracy on the given data [42]. From the list of classifiers used in this work and considering the implementation of AdaBoost in scikit-learn library of Python, we decided to use the Extra-Trees classifier as a weak classifier for running AdaBoost. This algorithm will first fit the Extra-Trees model on the original dataset and then fit the additional copies of the classifier in such a way that the these classifiers focus more on the incorrectly classified instances [43]. The function used here is `AdaBoostClassifier()`.

4.5 Validation Schemes

Validation schemes are used to determine how accurate the predictions made by the models are. We used two different cross validation (CV) schemes:

***n*-fold Cross Validation:** In this strategy, the given dataset is partitioned into n equal subsets. Each subset is once used as a validation dataset and remaining $n-1$ subsets form a training dataset. This process is repeated n times and the overall performance is calculated by computing the average over all n runs. In this work, we have used three-fold CV for model selection and ten-fold CV for assessing performance of the models.

Leave One Subject Out Cross Validation (LOSOCV): LOSOCV is similar to n-fold cross validation. Instead of splitting the data arbitrarily into n subsets, the data are grouped by p subjects. This strategy helps identifying how well the data from $p-1$ different subjects are capable of classifying the data of the singled out subject.

5 Evaluation Approach

The general classification approach, we follow in this work, consists of the following steps: First, a feature set is computed. Feature sets can consist of any combination of the features described in Sect. 3.2. Second, a feature selection step is carried out. Third, we select the best performing models for each feature type based on the outcome of a three-fold CV on $\mathcal{DB}_{\text{partial}}$. Using this information we perform classification on $\mathcal{DB}_{\text{full}}$. There are small differences between the global classification approach and the personalized classification approach, which are explained in this section. Figure 2 outlines the overall approach.

5.1 Global Classifiers

In this strategy, feature computation and selection are performed separately for $\mathcal{DB}_{\text{partial}}$ and $\mathcal{DB}_{\text{full}}$. Model selection is performed on $\mathcal{DB}_{\text{partial}}$ and the information obtained from this step is used for classification of $\mathcal{DB}_{\text{full}}$. The steps are explained in detail below:

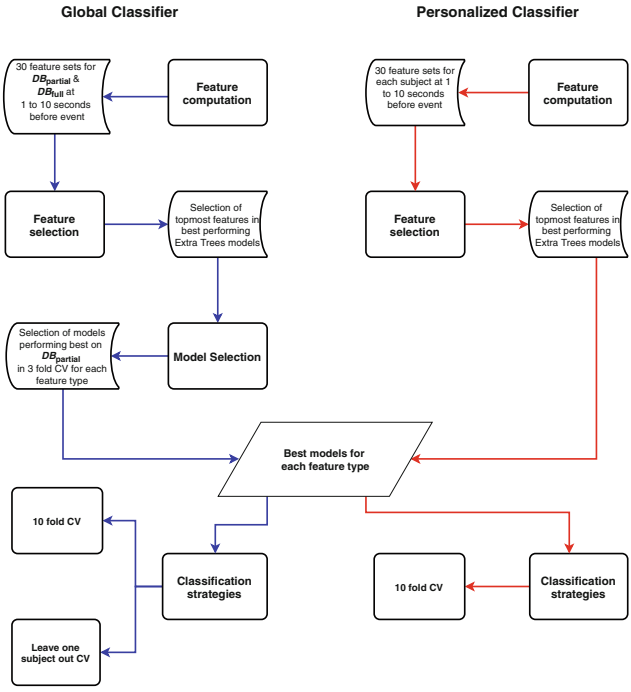

Fig. 2. Flowchart of the complete classification approach. Blue arrows mark the flow for global classifiers, red arrows represent the personalized classifiers. (Color figure online)

Feature Computation: Raw, statistical, Fourier, and angular features are computed for all specified segment lengths. Additionally, we generate more feature types by considering all possible combinations of the four features mentioned above. For each of these feature sets, we compute the PC projection and use them as another feature set. Hence there are 30 feature sets for each segment length.

Feature Selection: This step is based on the Extra-Trees classifier. For each feature set, we perform a ten-fold CV on all possible combinations of parameters for Extra-Trees models. In each validation step, we store each model's performance (accuracy) and feature importances. Feature importance reflects a feature's relevance in the classification. Based on each model's highest average accuracy, we select the best models. From these we compute the union of the previously saved topmost features. These features are used in our subsequent analysis.

SML Models: The feature sets used in this step are the ones that have been obtained after feature selection step.

1. Model Selection - Since there are a wide variety of possible models it is better to start with the $\mathcal{DB}_{\text{partial}}$ for identifying the best models for each feature set. For each feature set, we perform three-fold CV with all possible models. We compute performance measures for assessing the performance of classifiers.

We then determine the best models by considering the best values for each performance measure keeping seconds, features constant. We generate a list of models that have been identified as best with specific feature sets and save them in **SelectedModels**.

2. For $\mathcal{DB}_{\text{full}}$, we perform ten-fold CV using the information in **SelectedModels**. We compute performance measures and determine the highest accuracy for each feature set. We also perform LOSOCV for each subject in $\mathcal{DB}_{\text{full}}$.

Validation of Model Selection: To validate our model selection step, we compared the average ratio of misclassified samples during three-fold CV for $\mathcal{DB}_{\text{partial}}$ with the average ratio when performing ten-fold CV on $\mathcal{DB}_{\text{full}}$. It turned out, that performance is comparable (e.g., 0.24 for three-fold CV on $\mathcal{DB}_{\text{partial}}$ and 0.25 for ten-fold CV on $\mathcal{DB}_{\text{full}}$). Hence, it is reasonable to infer model and feature combination that would work the best for the $\mathcal{DB}_{\text{full}}$ dataset by using information from $\mathcal{DB}_{\text{partial}}$.

5.2 Personalized Classifiers

In this strategy, classification is performed separately for each subject which has at least ten samples and further contains at least three samples of each, hinging, standing, and sitting. The complete workflow is shown in Fig. 2. There are small differences in this approach compared to the global classifier approach. Here, in **feature computation**, the 30 feature sets are computed for each subject separately instead of computing feature sets for all samples in the dataset together. **Feature selection** is performed in a similar manner as in Sect. 5.1. Model and feature combination is determined from **SelectedModels** which had been generated in Sect. 5.1. This information is used to perform classification using **ten-fold CV** on feature sets obtained after feature selection.

6 Results

In this section, we present the main results obtained through global and personalized classifier approaches.

6.1 Global Classification Results

In global classification, ten-fold CV on $\mathcal{DB}_{\text{full}}$ resulted in highest average classification accuracy values between 0.61 and 0.80 depending on the feature sets. The highest average classification accuracy was achieved using PC of statistical feature set, while the lowest average classification accuracy was obtained with PC of angular feature set. Various feature sets performed well with an average classification accuracy of 0.79.

For PC of statistical feature set, the highest average classification accuracy of 0.80 was achieved by using eight specific AdaBoost models and two specific Extra-Trees models at a window size of $s = 1\,\text{s}$ before the event. Table 3 (a) shows values of the respective confusion matrix. Figure 3 shows the average classification accuracy and standard deviation of several specific models for window

sizes ranging from $s = 1\,\text{s}$ to $s = 10\,\text{s}$ for ten-fold CV on $\mathcal{DB}_{\text{full}}$. We found that all models performed best at a window size of one second. Considering longer motion sequences did not improve classification results in this scenario.

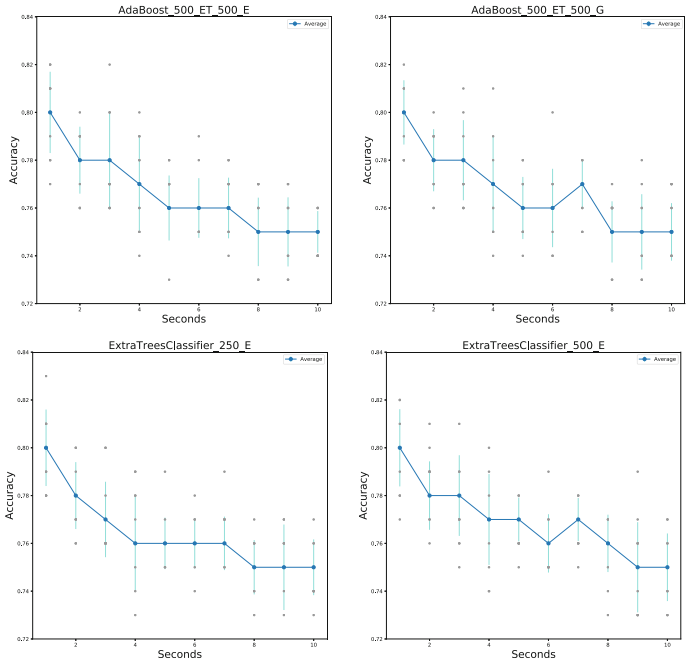

Fig. 3. Average classification accuracy of ten-fold CV for $\mathcal{DB}_{\text{full}}$ (blue dots and lines) with standard deviation (vertical cyan lines) using PC of Statistical feature set generated at $s = 1, 2, \ldots, 10\,\text{s}$ before the event. Grey dots mark the accuracy value in each fold of the ten-fold CV. The title of each plot contains the classifier and the values of the parameters used to train that model. (Color figure online)

In ten-fold CV, samples for training and validation are taken randomly. Thus, we cannot derive concrete conclusions how well the classification works for a new, unknown user. To this end, we performed LOSOCV. In this scenario, all subjects have highest classification accuracy greater than 0.50 i.e. with different feature sets and model combinations it is possible to classify each subject with accuracy at least more than 0.50. 65% and 46% of the total number of subjects have highest classification accuracy of over 0.80 and 0.90 respectively. 37% of the total number of subjects reached a classification accuracy of 1. Figure 4 shows a plot representing the percentage of subjects which have accuracy x or more where $x = 0.0, 0.1, \ldots, 1.0$. Table 3 (b) shows the confusion matrix at a window size of $s = 1\,\text{s}$. This quality was achieved which for 32% of the subjects.

6.2 Personalized Classification

Samples belonging to 41 different subjects in the $\mathcal{DB}_{\text{full}}$ dataset satisfied the requirements for personalized analysis described in Sect. 5.2. Hence, they have

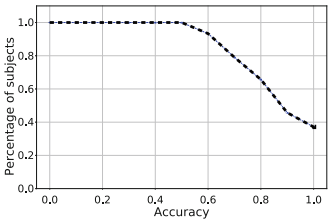

Fig. 4. Percentage of subjects with an average classification accuracy of x or more for $x = 0.0, 0.1, ..., 1.0$ in $\mathcal{DB}_{\text{full}}$. On the vertical axis 1.0 represents 100%.

Table 2. List of features and classifiers including used parameter values which performed best in the classification.

Feature	Classifier	Value of parameter
Angular	AdaBoost	$e = 500$ & $c =$ Extra Trees with $e = 500$ & Entropy
Raw	ANN	$h_i = 900$, $f_{\text{act}} = l$, $f_{\text{opt}} = s$ & $l = a$
PC of statistical	ANN	$h_i = 600$, $f_{\text{act}} = l$, $f_{\text{opt}} = s$ & $l = a$
Raw & angular	ANN	$h_i = 800$, $f_{\text{act}} = l$, $f_{\text{opt}} = s$ & $l = a$
PC of statistical & angular	ANN	$h_i = 300$, $f_{\text{act}} = l$, $f_{\text{opt}} = s$ & $l = a$
PC of statistical & angular & raw	ANN	$h_i = 300$, $f_{\text{act}} = l$, $f_{\text{opt}} = s$ & $l = a$
PC of statistical & angular & Fourier	ANN	$h_i = 600$, $f_{\text{act}} = l$, $f_{\text{opt}} = a$ & $l = i$

Table 3. Confusion matrix values for global and personalized classification approaches. HI abbreviate hinging, SI sitting, and ST standing. Sub-tables (a) and (b) contain results from the global classifier. In (a), ten-fold CV was used, in (b) LOSCOV. Subtable (c) and (d) show two exemplary results from the personalized classifier.

	(a)			(b)			(c)			(d)		
True & Predicted	HI	SI	ST	HI	SI	ST	HI	SI	ST	HI	SI	ST
HI	0.93	0.02	0.01	1	0	0	0.99	0.00	0.01	1.00	0.00	0.00
SI	0.04	0.73	0.21	0	1	0	0.00	0.72	0.18	0.00	1.00	0.03
ST	0.03	0.25	0.78	0	0	1	0.01	0.28	0.81	0.00	0.00	0.97

been considered for further per subject analysis. We performed ten-fold CV for each of these subjects and calculated the average accuracy for each subject. From these average accuracies the models with highest average accuracies were identified keeping model category, feature sets, segment length and subject constant. The obtained models were further reduced by selecting the models having highest average accuracies at all segment lengths for a given subject and feature set. A detailed specification of these models is given in Table 2. The average accuracy over subjects for the models in Table 2 lies between 0.70 and 0.85. Figure 5 shows average accuracy and standard deviation plots of four such models. Tables 3 (c) and (d) show the confusion matrix obtained for a subject with

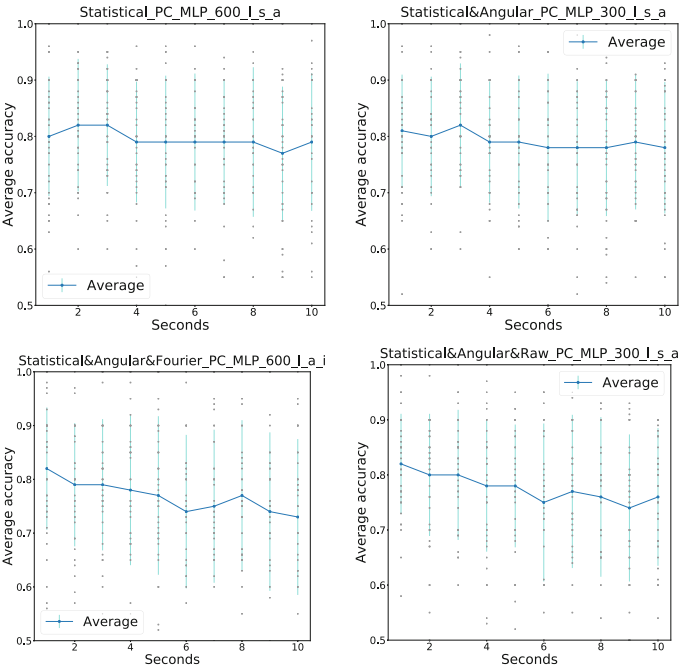

Fig. 5. Average classification accuracy of the four best performing ANN models for all subjects and at all time ranges on $\mathcal{DB}_{\text{full}}$.

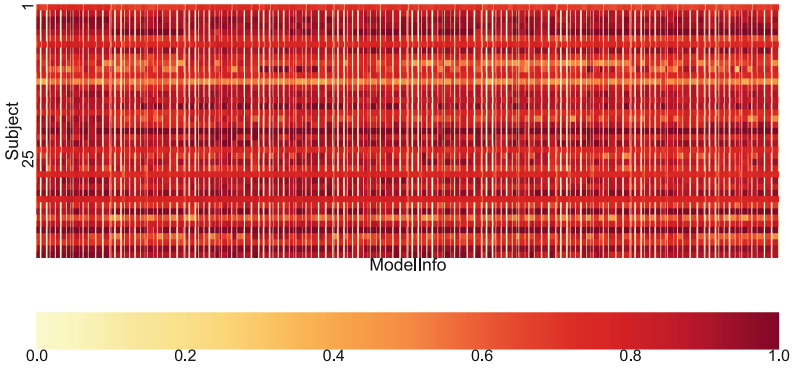

Fig. 6. Heatmap of the average accuracy for $\mathcal{DB}_{\text{full}}$. Subjects are on the y-axis and feasible combinations of PC feature, time range and model on the x-axis.

average classification accuracy of 84% and 98% respectively. By using a certain set of features and models it is possible to classify the postures of 15 subjects with 100% average classification accuracy resulting in an ideal confusion matrix. To get an overall idea of the performance of different models and features for the 41 subjects, Fig. 6 shows a heatmap of average classification accuracies. Each cell in the heatmap denotes the average accuracy obtained for a given subject on the

y-axis and model as well as feature combination on the x-axis. The horizontal view of the heatmap shows that there are certain subjects which have accuracies of over 50% in almost all possible model and feature combinations. The vertical view shows that certain model and feature combinations fail to accurately classify any subject.

7 Conclusion and Future Work

In this work, we presented a series of approaches for classification of a user's position, based on posture readings of a wearable device. We tested various feature sets and classifiers using global and personalized classification approaches.

It turned out that a personalized classification approach can better predict the position than a global approach if sufficient data is available for training of such personalized models. We found that 65% of the subjects have an average accuracy of larger than 0.8 for global classification. In our case, it was feasible to perform personalized analysis on 38% of total number of subjects. We observed that 97% of these subjects reached an average accuracy of over 0.8 and for 68% the average accuracy climbed to over 0.9. Hence, it is possible to come up with personalized posture classification based on the *Gokhale SpineTracker™* readings, even if only few samples are given per subject and position.

This classification can be further extended to differentiate between good and bad postures so as to notify the user whenever they are in the wrong posture for a particular task. Additionally, allowing a user to train custom positions and adding these into classification is of interest and challenging when only few data points are present. Further extension is also possible by modifying the sensors to improve the performance of the automated system. The accelerometer sensors can be further developed into inertial sensors by extending them with a gyroscope. This would provide more information and potentially improve the classification of postures.

References

1. Vos, T., Abajobir, A.A., Abate, K.H., Abbafati, C., Abbas, K.M., et al.: Global, regional, and national incidence, prevalence, and years lived with disability for 328 diseases and injuries for 195 countries, 1990–2016: a systematic analysis for the global burden of disease study 2016. The Lancet **390**(10100), 1211–1259 (2017)
2. NINDS: Low back pain fact sheet (2017). https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Low-Back-Pain-Fact-Sheet/. Accessed 16 May 2018
3. Enterprises, G.M.: Gokhale spinetracker. https://gokhalemethod.com/. Accessed 23 July 2018
4. Noiumkar, S., Tirakoat, S.: Use of optical motion capture in sports science: a case study of golf swing. In: 2013 International Conference on Informatics and Creative Multimedia, pp. 310–313. IEEE (2013)
5. MacIver, M., Sharabash, N., Nelson, M.: Prey-capture behavior in gymnotid electric fish: motion analysis and effects of water conductivity. J. Exp. Biol. **204**(3), 543–557 (2001)

6. Culhane, K.M., O'Connor, M., Lyons, D., Lyons, G.M.: Accelerometers in rehabilitation medicine for older adults. Age Ageing **34**(6), 556–560 (2005)
7. Merriaux, P., Dupuis, Y., Boutteau, R., Vasseur, P., Savatier, X.: A study of vicon system positioning performance. Sensors **17**(7), 1591 (2017)
8. Riaz, Q., Guanhong, T., Krüger, B., Weber, A.: Motion reconstruction using very few accelerometers and ground contacts. Graph. Models **79**, 23–38 (2015)
9. Slyper, R., Hodgins, J.K.: Action capture with accelerometers. In: Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation. SCA 2008, pp. 193–199. Eurographics Association (2008)
10. Vlasic, D., et al.: Practical motion capture in everyday surroundings. ACM Trans. Graph. **26**(3), 35 (2007)
11. Farella, E., Benini, L., Riccò, B., Acquaviva, A.: MOCA: a low-power, low-cost motion capture system based on integrated accelerometers. Adv. Multimedia **2007**(1), 1 (2007)
12. Weise, T., Bouaziz, S., Li, H., Pauly, M.: Realtime performance-based facial animation. ACM Trans. Graph. **30**(4), 77:1–77:10 (2011)
13. Cao, C., Bradley, D., Zhou, K., Beeler, T.: Real-time high-fidelity facial performance capture. ACM Trans. Graph. **34**(4), 46:1–46:9 (2015)
14. Hoffmann, J., Brüggemann, B., Krüger, B.: Automatic calibration of a motion capture system based on inertial sensors for tele-manipulation. In: 7th International Conference on Informatics in Control, Automation and Robotics, June (2010)
15. Ma, C.Z.H., Ling, Y.T., Shea, Q.T.K., Wang, L.K., Wang, X.Y., Zheng, Y.P.: Towards wearable comprehensive capture and analysis of skeletal muscle activity during human locomotion. Sensors **19**(1), 195 (2019)
16. Zhao, W., Chai, J., Xu, Y.Q.: Combining marker-based mocap and RGB-D camera for acquiring high-fidelity hand motion data. In: Proceedings of ACM SCA, pp. 33–42 (2012)
17. Stollenwerk, K., Vögele, A., Krüger, B., Hinkenjann, A., Klein, R.: Automatic temporal segmentation of articulated hand motion. In: Gervasi, O., et al. (eds.) ICCSA 2016. LNCS, vol. 9787, pp. 433–449. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42108-7_33
18. Wan, C., Probst, T., Van Gool, L., Yao, A.: Dense 3D regression for hand pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5147–5156 (2018)
19. Williams, J.M., Haq, I., Lee, R.Y.: Dynamic measurement of lumbar curvature using fibre-optic sensors. Med. Eng. Phys. **32**(9), 1043–1049 (2010)
20. Stollenwerk, K., Müllers, J., Müller, J., Hinkenjann, A., Krüger, B.: Evaluating an accelerometer-based system for spine shape monitoring. In: Gervasi, O., et al. (eds.) ICCSA 2018. LNCS, vol. 10963, pp. 740–756. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-95171-3_58
21. Xu, W., et al.: MonoPerfCap: human performance capture from monocular video. ACM Trans. Graph. **37**(2), 27:1–27:15 (2018)
22. Alldieck, T., Magnor, M., Xu, W., Theobalt, C., Pons-Moll, G.: Detailed human avatars from monocular video. In: International Conference on 3D Vision, pp. 98–109. IEEE (2018)
23. Iqbal, U., Doering, A., Yasin, H., Krüger, B., Weber, A., Gall, J.: A dual-source approach for 3D human pose estimation from single images. Comput. Vis. Image Underst. **172**, 37–49 (2018)
24. Dabral, R., Mundhada, A., Kusupati, U., Afaque, S., Sharma, A., Jain, A.: Learning 3D human pose from structure and motion. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11213, pp. 679–696. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01240-3_41

25. Bernard, J., Dobermann, E., Vögele, A., Krüger, B., Kohlhammer, J., Fellner, D.: Visual-interactive semi-supervised labeling of human motion capture data. In: Visualization and Data Analysis, January (2017)
26. Baumann, J., Wessel, R., Krüger, B., Weber, A.: Action graph: a versatile data structure for action recognition. In: GRAPP 2014 - International Conference on Computer Graphics Theory and Applications, SCITEPRESS, January (2014)
27. Riaz, Q., Vögele, A., Krüger, B., Weber, A.: One small step for a man: estimation of gender, age, and height from recordings of one step by a single inertial sensor. Sensors **15**(12), 31999–32019 (2015)
28. Consmüller, T., et al.: Automatic distinction of upper body motions in the main anatomical planes. Med. Eng. Phys. **36**(4), 516–521 (2014)
29. Jeyhani, V., Mahdiani, S., Viik, J., Oksala, N., Vehkaoja, A.: A novel technique for analysis of postural information with wearable devices. In: IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks, March, 30–33 (2018)
30. Wong, W.Y., Wong, M.S.: Trunk posture monitoring with inertial sensors. Eur. Spine J. **17**(5), 743–753 (2008)
31. Voinea, G.D., Butnariu, S., Mogan, G.: Measurement and geometric modelling of human spine posture for medical rehabilitation purposes using a wearable monitoring system based on inertial sensors. Sensors **17**(1), 3 (2017)
32. Fathi, A., Curran, K.: Detection of spine curvature using wireless sensors. J. King Saud Univ.-Sci. **29**(4), 553–560 (2017)
33. Cajamarca, G., Rodríguez, I., Herskovic, V., Campos, M., Riofrío, J.C.: StraightenUp+: monitoring of posture during daily activities for older persons using wearable sensors. Sensors **18**(10), 3409 (2018)
34. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
35. Tarca, A.L., Carey, V.J., Chen, X., Romero, R., Drăghici, S.: Machine learning and its applications to biology. PLoS Comput. Biol. **3**(6), e116 (2007)
36. Scikitlearn: Nearest neighbors. http://scikit-learn.org/stable/modules/neighbors.html. Accessed 19 Sept 2018
37. Bhatia, N.: Vandana: Survey of nearest neighbor techniques. CoRR abs/1007.0085 (2010)
38. Lan, K., Wang, D.T., Fong, S., Liu, L.S., Wong, K.K., Dey, N.: A survey of data mining and deep learning in bioinformatics. J. Med. Syst. **42**(8), 139:1–139:20 (2018)
39. Ho, T.K.: Random decision forests. In: Third International Conference on Document Analysis and Recognition, vol. 1, pp. 278–282 (1995)
40. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Mach. Learn. **63**(1), 3–42 (2006)
41. Tu, J.V.: Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes. J. Clin. Epidemiol. **49**(11), 1225–1231 (1996)
42. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
43. Hastie, T., Rosset, S., Zhu, J., Zou, H.: Multi-class adaboost. Stat. Interface **2**(3), 349–360 (2009)