Human Activity Recognition using Sensor Data

Bhanu Jain, Ronny Mathew

January 29, 2019

Abstract

We are in an age of sensors where almost every device we interact with collects megabytes of physiological data every minute. Our goal in this project is to find the best classifier that creates a model from sensor data and predicts the activity that's currently being performed by the person. While there is a lot of work done around activity recognition, our motive here is to simplify raw data and reduce the number of input features and using the best classifier possible to accurately predict and further enhance this model to performing a real world situation.

1 Introduction

With the endless possibilities of collecting data the goal here is to formulate a model that can be used in multi-domain settings, where we can train our model on one domain and predict on other. To begin with, we will use an existing dataset to train a model accurately. One we achieve desired results we will move this model to a real time environment, bridging the domain gap through feature selection and pre-processing.

2 Data Set

We used the "Human Activity Recognition Using Smartphones Data Set" provided by UCI. The experiments were carried out with a group of 30 volunteers within an age bracket of 19-48 years. They performed a protocol of activities composed of six basic activities: three static postures (standing, sitting, lying) and three dynamic activities (walking, walking downstairs and walking upstairs). The experiment also included postural transitions that occurred between the static postures. These are: stand-to-sit, sit-to-stand, sit-to-lie, lie-to-sit, stand-to-lie, and lie-to-stand. All the participants were wearing a smartphone (Samsung Galaxy S II) on the waist during the experiment execution. We captured 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50Hz using the embedded accelerometer and gyroscope of the device. The experiments were video-recorded to label the data manually. The obtained dataset was randomly partitioned into two sets, where 70% of the volunteers was selected for generating the training data and 30% the test data.

The sensor signals (accelerometer and gyroscope) were pre-processed by applying noise filters and then sampled in fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window). The sensor acceleration signal, which has gravitational and body motion components, was separated using a Butterworth low-pass filter into body acceleration and gravity. The gravitational force is assumed to have only low frequency components, therefore a filter

with 0.3 Hz cutoff frequency was used. From each window, a vector of 561 features was obtained by calculating variables from the time and frequency domain.

2.1 Attribute Information

The dataset is then divided in two parts and they can be used separately.

- Inertial sensor data
  - Raw triaxial signals from the accelerometer and gyroscope of all the trials with with participants.
  - The labels of all the performed activities.
- Records of activity windows. Each one composed of:
  - A 561-feature vector with time and frequency domain variables.
  - Its associated activity label.
  - An identifier of the subject who carried out the experiment.

For the scope of this project, we are only using pre-processed data, however we plan to use to raw data and do a domain shift on the learned classifier for live classification.

2.1.1 Labels

There are totals 12 labels as mentioned before. Thease are WALKING, WALKING_UPSTAIRS, WALKING_DOWNSTAIRS, SITTING, STANDING, LAYING, STAND_TO_SIT, SIT_TO_STAND, SIT_TO_LIE, LIE_TO_SIT, STAND_TO_LIE, LIE_TO_STAND

3 Data Preprocessing

Before passing the input data to a classifier we performed multiple operations on the data that would improve the accuracy of the prediction. These operations can be divided into:

3.1 Feature Scaling

The selected features were scaled using a Standard Scalar. Scaling the features resulted in zero mean and unit variance across the data. On the normalized data we did a random shuffling to reduce overfitting, which would help us choose a better classifier. The results in Table 1 were obtained without any feature selection. Seeing good results with SVMs suggest that the data is already optimized and PCA would not have much impact on the results. We chose to shuffle our data for any further experiments as it gave better accuracy.

Table 1: Classification with normalization and random seed

	precision	recall	accuracy
linear-support-vector-machine	0.953	0.952	0.952
linear-discriminant-analysis	0.950	0.949	0.949
rbf-support-vector-machine	0.916	0.918	0.918
random-forest	0.887	0.886	0.886
knn	0.890	0.885	0.885
poly-support-vector-machine	0.858	0.873	0.873
quadratic-discriminant-analysis	0.817	0.847	0.847
decision-trees	0.810	0.807	0.807
gaussian-naive-bayes	0.794	0.747	0.747

Table 2: Classification with normalization and no shuffle

	precision	recall	accuracy
linear-support-vector-machine	0.953	0.953	0.953
linear-discriminant-analysis	0.950	0.949	0.949
rbf-support-vector-machine	0.918	0.917	0.917
knn	0.890	0.885	0.885
random-forest	0.880	0.876	0.876
poly-support-vector-machine	0.872	0.867	0.867
quadratic-discriminant-analysis	0.809	0.843	0.843
decision-trees	0.810	0.806	0.806
gaussian-naive-bayes	0.794	0.747	0.747

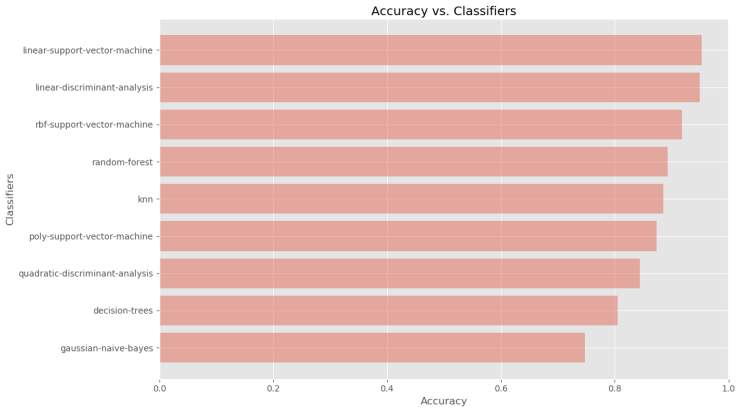

Figure 1: Accuracy vs Classifier with normalization and random seed

3.2 Feature Selection

The input dataset has large number of features, 561 to be exact. We evaluated the performance of multiple classifiers by taking multiple values of PCA components. Using PCA we were able to reduce the number of features passed to the classifiers.

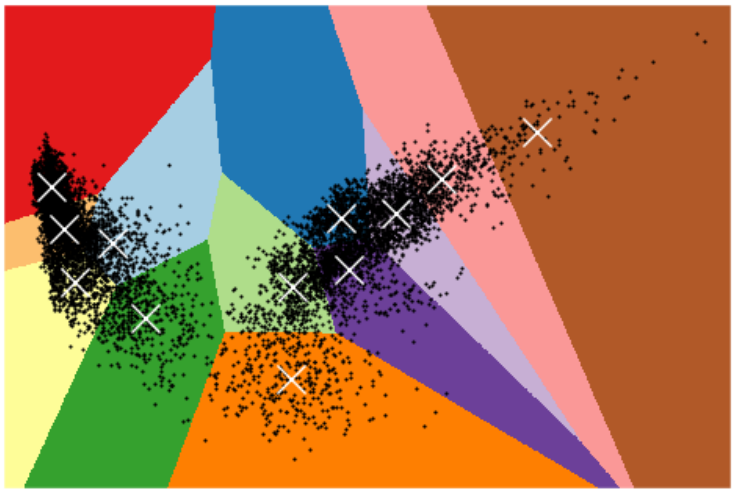

Figure 2: k-means performed on 2 PCA components

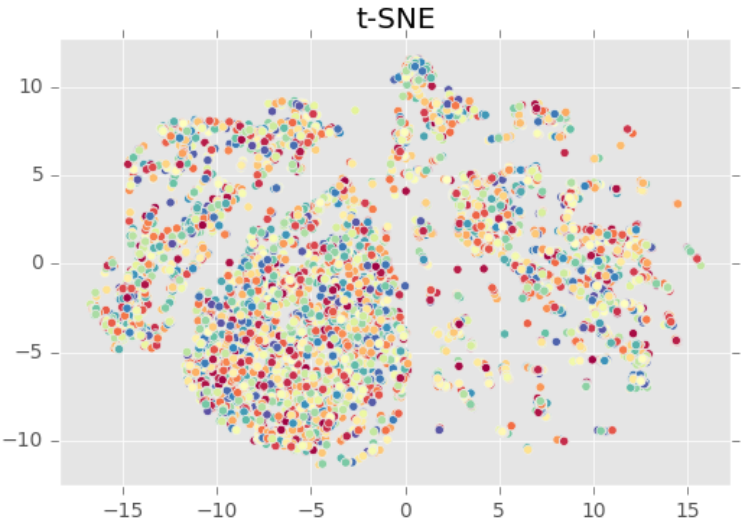

Figure 3: Distribution of data with t-SNE and 2 PCA components

Figure 2 shows k-means on a 2 component PCA . We can observe clear linear boundaries between all our 12 labels, but the data points are overlapping. With our initial data analysis we wanted to reduce the number of features to have a better performance and reduction in complexity. We ran a PCA with multiple components and found though there was some improvement in performance, but the accuracy was almost the same. Thus it is safe to say that the input data has optimal variance across the normal. In Figure 4 we can see the most of the classifiers give constant performance over increased PCA components. It is also worth mentioning that the accuracy also reduces as compared to Table 1, thus there is a loss of information while applying PCA.

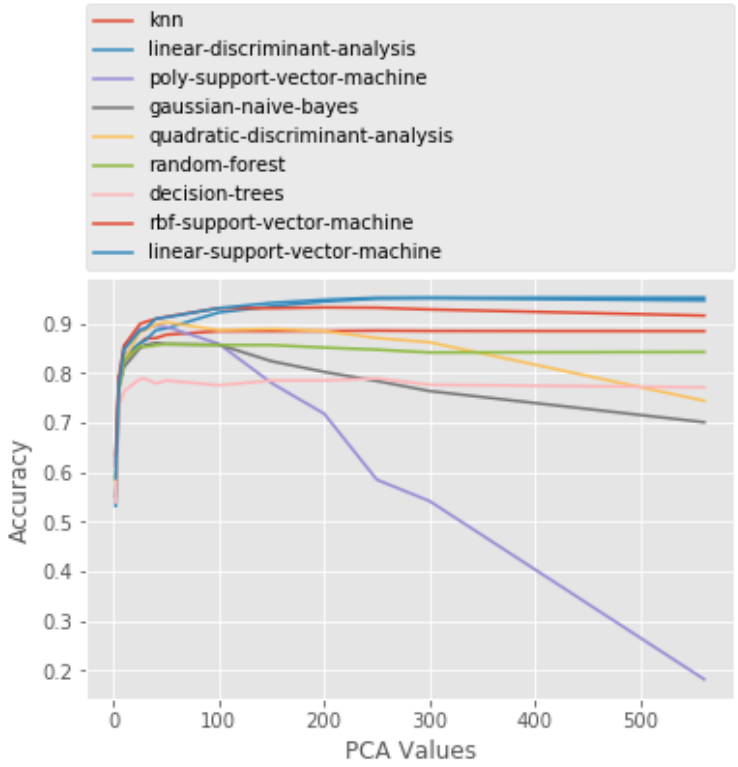

Figure 4: Comparison of PCA on multiple classifiers

3.3 Cross Validation

When there are certain hyper parameters that needs to be computed for the model to perform efficiently for the given data we use Cross Validation to find these parameters. We used Grid Search Cross validation to performs a grid search on the available data and list of hyper parameters to find the ones that gave the best prediction.

4 Classification

Once the preprocessing was completed, we needed a classifier to predict the activity being performed. We evaluated multiple classifiers to find the best fit for our classification task. These are the classifiers we considered for the classification step (sorted in order of decreasing accuracy)

4.1 SVM

The best prediction was provided by the SVM model. The SVM model uses a kernel to find the similarity and model the data.

$$argmin_{e,b,\xi\geq 0}\frac{1}{2}w^T w + C\Sigma_{i=1}^{m}\xi_i$$
$$\text{s.t. } y_i(w^T x_i + b) \geq 1 - \xi_i, \text{ i=1,..,m}$$

We used grid search cross validation to find the best parameters to use for SVM like C and gamma (for RBF kernel).

4.2 Linear Discriminant Analysis

Linear Discriminant Analysis finds a linear combination of features that will separate and represent the classes of the activities. LDA is similar to PCA where both of them tries to find a linear combination of features that will represent the input data.

$$p(x \mid y = k) = N(x; \mu_k, \Sigma)$$
$$\text{where } \mu_k = \frac{1}{N_k}\Sigma_{n:y_n=k} x_n$$
$$\text{and } \Sigma_k = \frac{1}{1/N_k}\Sigma_{n:y_n=k}(x_n - \mu_k)^T(x_n - \mu_k)$$

We have used the default SVD solver of Sci-kit learn to perform Classification and prediction.

4.3 Random Forests

Random Forests are a special case of bagging of decision trees in which we have multiple number of trees. Each tree in the forest is passed a subset of the features and a subset of the data points with replacement. We got comparable results from random forest in our application.

4.4 k Nearest Neighbors

k Nearest Neighbor is a non-generalizing classification method where we do not create a model with the training data but simply store it to get the nearest neighbor majority count of the test data and assigns the class to it. In our application, we noticed that kNN gave nearly equivalent prediction to Random Forests but took more time to run.

4.5 Quadratic Discriminant Analysis

Quadratic Discriminant Analysis are similar to LDA expect the QDA has a quadratic decision boundary. QDA did not provide a better prediction than LDA or any of the above methods.

4.6 Decision Trees

Decision trees infers simple decision rules by going through the input data and these rules are used to predict the output. Decision trees provided lower predictions for our application when used with the entire dataset. We also observed that the accuracy improved a little while using PCA to reduce the number of input features to the classifier.

4.7 Naive Bayes

Naive Bayes with a Gaussian prior provided the worst performance in our application. This could be because Naive Bayes assumes Independence between the features and the features are in fact related. For example, we have the x, y and z values of the Accelerometer and Gyrometer, the activity will depend on the values of these parameters.

$$p(C_k|x_1, \ldots, x_n) = \frac{1}{Z}p(C_k)\prod_{i=1}^{n} p(x_i|C_k) \text{ where } \hat{y} = C_k \text{ for some k}$$

5 Evaluation Metrics

We conducted a couple of experiments and evaluated each under four metrics listed below

5.1 Accuracy

Accuracy is the measure of the test data points correctly classified. It measures the true positives and true negatives out of the total test data points.

$$\text{Accuracy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}}$$

5.2 Precision

Precision is the measure of obtaining the accurate positive values from the test data. It measures the true positives from the actual positive values in the test data.

$$\text{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}}$$

5.3 Recall

Recall is the measure of sensitivity of the classifier. It measures the true positives out of the true values of the test data.

$$\text{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}}$$

5.4 F-1 Score

F1 score is the weighted average of the precision and recall and in multi-class classification it is the weighted average of the F1 score of each class. The best value of F1 score we can get is 1 and worst is 0.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

6 Experiment Results

Our experiments were conducted in three stages which were process the data, select a classifier and optimize. With feature selection and scaling we found that the data was insensitive to PCA. Though there was some improvement in performance, but the accuracy also suffered. We got better results with normalizing and shuffling the data.

In the second stage, while running the test on multiple classifiers as mentioned in section 4 we found that SVMs performed better than other classifiers. Moreover 'linear' kernel $\langle x, x' \rangle$ had a better performance over 'poly' $(\gamma \langle x, x' \rangle + r)^d . d$ and 'rbf' $\exp(-\gamma |x - x'|^2)$. All the three kernels in SVMs were tested using default hyperparameters i.e. regularization constant C = 1.0 and γ = 1 / 561 (total number features).

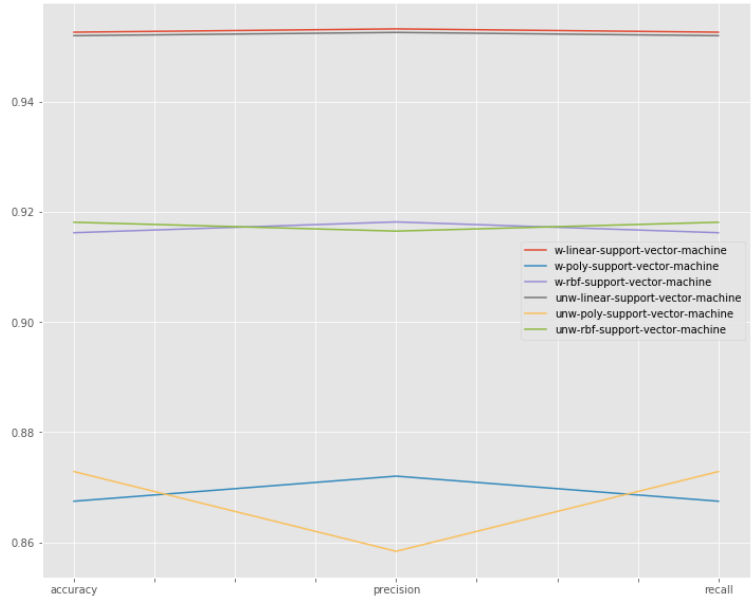

Figure 5: SVM Comparisons

Table 3: svm kernel performance

	precision	recall	accuracy	c	γ
linear-support-vector-machine	0.953	0.952	0.952	1.0	-
rbf-support-vector-machine	0.916	0.918	0.918	1.0	1/561
poly-support-vector-machine	0.858	0.873	0.873	1.0	1/561

Figure 5 show a comparison of SVM on the three kernels with weighted and non-weighted class. As we have a multi class data, weighted SVMs performs better than the non-weighted. Thus the default setting for further experiments will be with weights.

Table 3 shows that linear kernel has a better accuracy than the other two, however it is commonly known that non-linear kernels have the best possible predictive performance. While digging more into SVMs and kernels we found that non-linear kernels perform better over different set of hyperparameters. In order to achieve best possible set, we decided to optimize the hyperparameter for rbf and linear kernel using grid search.

Grid search is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm.Intuitively, the γ parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The γ parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. The C parameter trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors.

Table 4: svm kernel performance with optimal hyper parameters

	precision	recall	accuracy	c	γ
linear-support-vector-machine	0.953	0.952	0.952	1.0	-
rbf-support-vector-machine	0.954	0.953	0.953	1000.0	0.01

Table 4 shows the results after an exhaustive grid search on linear and rbf kernel. With a dual core machine, this analysis took more than an hour, as max iterations were set to -1 to prevent any inconsistency. With optional hyper parameters rbf kernel performs better than the linear kernel giving us an accuracy of 0.953 which is not only greater the linear kernel but outperforms any other classifiers we tested.

But looking at the comparable results by linear kernel, we are safe to say that if the number of features is large, we may not need to map data to a higher dimensional space. That is, the nonlinear mapping does not improve the performance. Using the linear kernel is good enough, and we only searches for the parameter C.

Thus, it is a trade-off better high precision and performance. Non linear kernels are assumed to perform better than linear, but linear kernel offer very high performance.

Table 5: Class Accuracy matrix for SVM with 'rbf' kernel

	precision	recall	f1-score	support
WALKING	0.96	0.99	0.97	496
WALKING_UPSTAIRS	0.94	0.97	0.95	471
WALKING_DOWNSTAIRS	0.99	0.94	0.97	420
SITTING	0.97	0.9	0.93	508
STANDING	0.92	0.98	0.95	556
LAYING	1	1	1	545
STAND_TO_SIT	0.95	0.83	0.88	23
SIT_TO_STAND	0.91	1	0.95	10
SIT_TO_LIE	0.7	0.72	0.71	32
LIE_TO_SIT	0.81	0.84	0.82	25
STAND_TO_LIE	0.71	0.71	0.71	49
LIE_TO_STAND	0.82	0.67	0.73	27
avg / total	0.95	0.95	0.95	3162

Table 5 illustrates how each class in our data set performed with SVM classifier with 'rbf' kernel. Note that these results were obtained with hyperparameters mentioned in tn Table 4. Transitional movement (which are STAND_TO_SIT, SIT_TO_STAND, SIT_TO_LIE, LIE_TO_SIT, STAND_TO_LIE, LIE_TO_STAND) giving a low f1-score (which uses precision and recall to compute accuracy) might be due to the fact that the sensors are sensitive to transitions and it is difficult to differentiate between minor change in orientation and acceleration. On the other hand, class WALKING with the highest support has a lower precision, which might indicate that the sensors are sensitive and erratic while doing more defined movements. Class LAYING with a perfect score of 1 also proves that when you remain in a state of no movement you can get better results from the sensors as compared to continuously moving (WALKING) which introduces noise in the data which in turn reduces our prediction accuracy.

Table 6: Confusion Matrix for SVM with 'rbf' kernel

Class Index	1	2	3	4	5	6	7	8	9	10	11	12
WALKING	489	5	2	0	0	0	0	0	0	0	0	0
WALKING_UP	15	455	0	0	0	0	1	0	0	0	0	0
WALKING_DOWN	5	20	395	0	0	0	0	0	0	0	0	0
SITTING	0	1	0	456	50	0	0	1	0	0	0	0
STANDING	0	0	0	9	547	0	0	0	0	0	0	0
LAYING	0	0	0	0	0	545	0	0	0	0	0	0
STAND_TO_SIT	0	1	0	2	0	0	19	0	0	0	1	0
SIT_TO_STAND	0	0	0	0	0	0	0	10	0	0	0	0
SIT_TO_LIE	0	0	0	0	0	0	0	0	23	0	9	0
LIE_TO_SIT	0	0	0	0	0	0	0	0	0	21	0	4
STAND_TO_LIE	0	0	0	2	0	2	0	0	10	0	35	0
LIE_TO_STAND	0	0	0	0	0	0	0	0	0	5	4	18

Table 6 illustrates error in classifying each class and is called the confusion or the error matrix. Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class. Thus the classifiers often fails to correctly classify similar activities such as WALKING, WALKING_UP, WALKING_DOWN

In order to analyze the performance over time, we conducted this last test, where we compared each classifier with their fit time and prediction time. The fit time was calculated over entire training set, while the predict time runs over the entire test set. Figure 6 shows the clear comparison of classifiers over time.

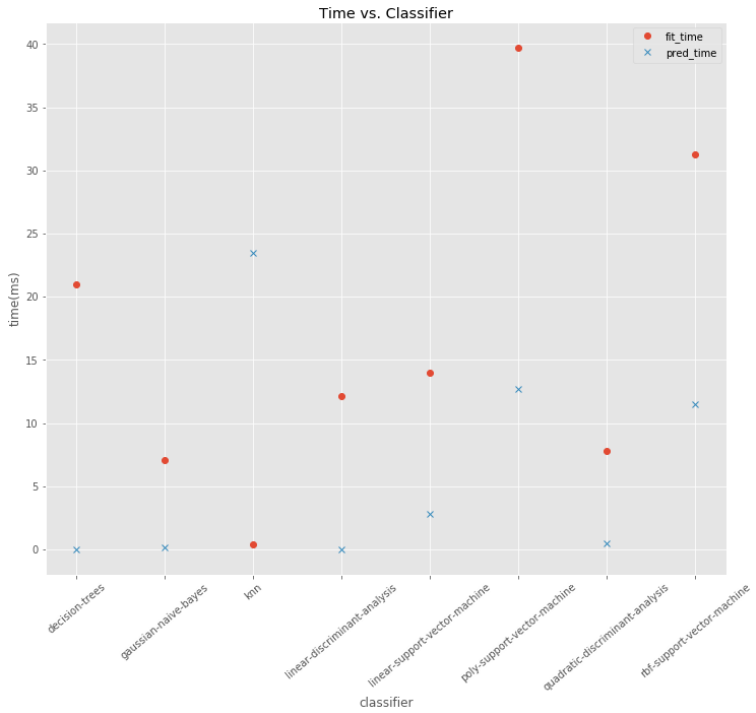

Figure 6: Time vs. Classifiers

Table 7: Classifiers Time Performance

	fit_time	pred_time
poly-support-vector-machine	41.142829	13.368196
rbf-support-vector-machine	32.850155	10.955805
decision-trees	22.099053	0.005529
linear-support-vector-machine	12.345608	2.803672
quadratic-discriminant-analysis	10.288529	0.446684
gaussian-naive-bayes	9.227992	0.186832
linear-discriminant-analysis	8.587444	0.010987
knn	0.369596	22.720753

Table 7 illustrates the time taken to fit and predict each classifier. With high accuracy of SVM 'linear' and 'rbf' kernel, it is now a choice of performance over time. In a situation where prediction time is of high importance, such as live tracking, choosing SVM with 'linear' kernel will be an optimal choice.

7 Extensions and Possibilities

With processed data, we are getting an accuracy of 0.953 which is good enough as a base. In future we propose to use this learned classifier for live predictions for continuous sensor inputs. This requires a domain shift as the data used to train is from old sensors and current sensor setting might be different from the earlier senors.

As the raw data is divided in time windows, we can do a time series modeling with a perfect memory context[7]. This might help in getting around the problem of domain shift, as the now we can map user actions as a time series and try to predict his next action.

Additional, we can combine this classifier with a clustering algorithm that could improve the performance and the accuracy of the prediction by grouping data into multiple clusters and then assign a label to the centroid using the classification. Thus, continuously improving the labels of learned data which would result in better accuracy.

The senor shortcomings in overlapping close activities can be rectified by using Kalman Filter[5] which uses past measurements to product precise estimates.Kalman filter is great tool to smooth signals and trajectories which produces reliable results and is widely used in guidance and navigation. We could use the algorithm to achieve better consistency is data for similar movements.

8 References

[1] CM Bishop, *Pattern recognition and machine learning*, Springer, 2006.

[2] Kevin P. Murphy, *Machine Learning: A Probabilistic Perspective, 1st Edition*, MIT Press, 2012.

[3] Wikipedia *Linear Discriminant Analysis*, Wikipedia, `https://en.wikipedia.org/wiki/Linear_discriminant_analysis` (Online).

[4] Scikit-learn *Scikit Learn Manual*, scikit-learn, `http://scikit-learn.org/stable/` (Online).

[5] Wikipedia *Kalman Filters*, Wikipedia, *`https://en.wikipedia.org/wiki/Kalman_filter`* (Online).

[6] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, *A Practical Guide to Support Vector Classification*, 2016. `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf` (Online)

[7] Tong Zang, *Perfect Memory Context Trees in time series modeling*, 2016. `https://arxiv.org/pdf/1610.08910v1.pdf` (Online)

[8] UCI, *Smartphone-Based Recognition of Human Activities and Postural Transitions*, UCI Datasets, `http://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions` (Online)

[9] Junkai Yan, *Human Activity Recognition*, 2015, `https://youtu.be/5Q6lR_1UYP8?t=659`