CS4801: PRINCIPLES OF MACHINE LEARNING 2018

Mini Project Report

Human Activity detection using Smartphones Data Set

Project under - Dr. Sahely Bhadra

Nikhil Kumar Yadav

Indian Institute of Technology, Palakkad
Computer Science And Engineering Department
Oct 29,2018

1 Data Set Description

This data set contains information obtained from 30 volunteers, within age bracket of 19-48 years, performing day to day activites like walking, walking upstairs,walking downstairs, sitting, standing and laying. The data is collected from the sensors of Samsung Galaxy S II mounted on the waists of the volunteers. The activites were manually labeled by watching videos of volunteers doing these tasks.

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 features was obtained by calculating variables from the time and frequency domain.

The features selected for this database come from the accelerometer and gyroscope 3-axial raw signals tAcc-XYZ and tGyro-XYZ. These time domain signals (prefix 't' to denote time) were captured at a constant rate of 50 Hz. Then they were filtered using a median filter and a 3rd order low pass Butterworth filter with a corner frequency of 20 Hz to remove noise. Similarly, the acceleration signal was then separated into body and gravity acceleration signals (tBodyAcc-XYZ and tGravityAcc-XYZ) using another low pass Butterworth filter with a corner frequency of 0.3 Hz.

Subsequently, the body linear acceleration and angular velocity were derived in time to obtain Jerk signals (tBodyAccJerk-XYZ and tBodyGyroJerk-XYZ). Also the magnitude of these 3-D signals were calculated using the Euclidean norm (tBodyAccMag, tGravityAccMag, tBodyAccJerkMag, tBodyGyroMag, tBodyGyroJerkMag).

Finally a Fast Fourier Transform (FFT) was applied to some of these signals producing fBodyAcc-XYZ, fBodyAccJerk-XYZ, fBodyGyro-XYZ, fBodyAccJerkMag, fBodyGyroMag, fBodyGyroJerkMag.

Then signals were then used to calculate their mean, std, meadian absolute deviation, max, min, signal magnitude area, energy, interquartile range, entropy, autorregresion coefficients, correlation, skewness, angle etc

This data set was pre-divided into training and test sets with traning set containing 7352 samples and test set containing 2947 samples.

2 Analytics

Fig1 shows that data points are somewhat equally distributed.

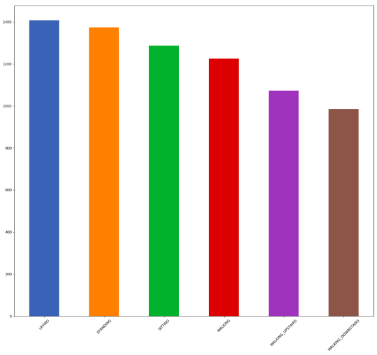

Figure 1: Distribution of class labels

I am using t-SNE to visualize the data points. t-Distributed Stochastic Neighbor Embedding is a non-linear dimensionality reduction algorithm used for exploring high-dimensional data. It's outputs provide better results than PCA and other linear dimensionality reduction models. This is because a linear method such as classical scaling is not good at modeling curved manifolds. It focuses on preserving the distances between widely separated data points rather than on preserving the distances between nearby data points. The algorithm computes pairwise conditional probabilities and tries to minimize the sum of the difference of the probabilities in higher and lower dimensions.

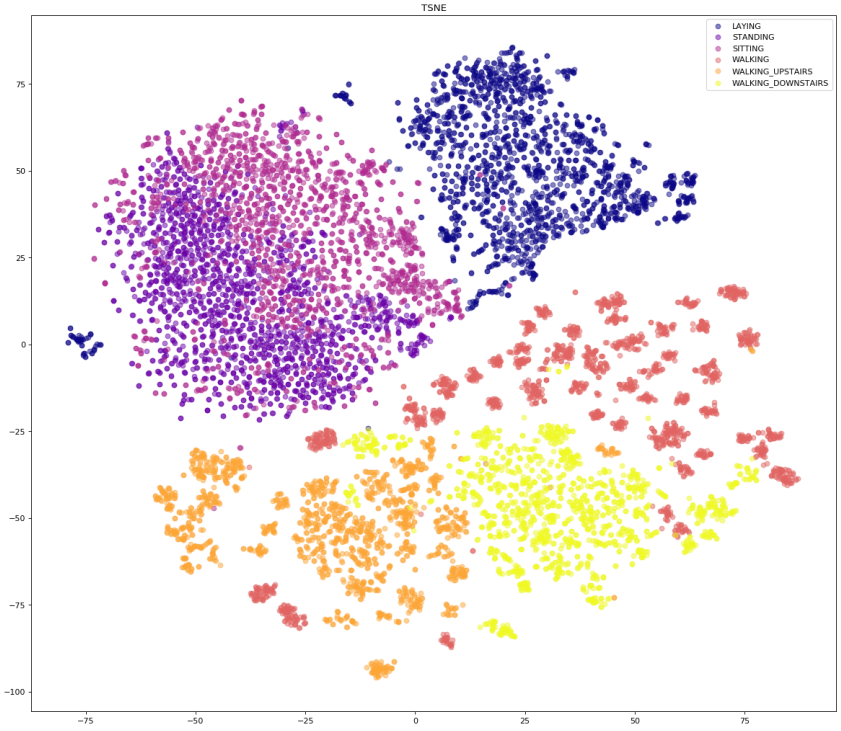

Figure 2: t-SNE visualization

From the figure it appears that somewhat that KNN should work best although it might not work better for class labels *STANDING* and *SITTING*.

3 Experimental Verification

I trained the data using the following models on two data sets one with original number of features(561) and the tsne transformed data.

- Decision Tree
- Random Forest
- KNN
- SVC
- Naive Bayes

The following results were obtained for original dataset -

	Model	Score
0	DecisionTreeClassifier	0.807601
1	RandomForestClassifier	0.920597
2	KNeighborsClassifier	0.903970
3	SVC	0.954869
4	GaussianNB	0.571089

The following results were obtained for original dataset -

	Model	Score
0	DecisionTreeClassifier	0.949515
1	RandomForestClassifier	0.962136
2	KNeighborsClassifier	0.943366
3	SVC	0.959871
4	GaussianNB	0.848220

Clearly by reducing the dimension we are not losing something. Hence it is better to transform the data two 2 dimensions and then train our model.

My hypothesis that KNN should work better is not clearly visible here as the labels *STANDING* and *SITTING* data points are nearby. Hence it will be better to make a our model by combining the power of different models.

So we will train the whole data using KNN and train the data points with labels *STANDING* and *SITTING* using SVC. Now while predicting our answer we will use KNN, if KNN predicts *STANDING* or *SITTING* then we will use SVC's prediction.

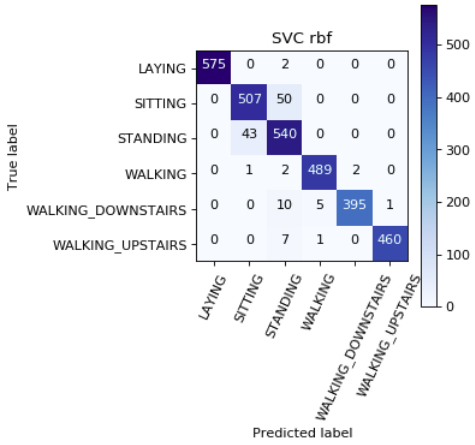

Figure 3: Confusion Matrix for SVC rbf

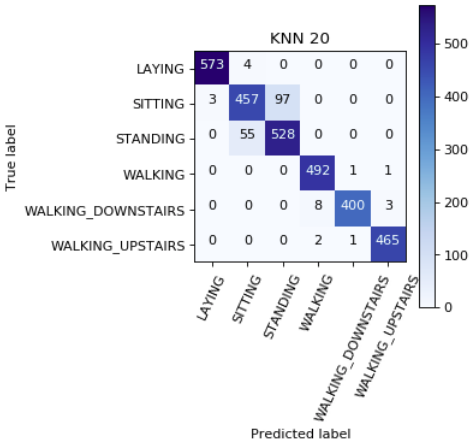

Figure 4: Confusion Matrix for KNN K=20

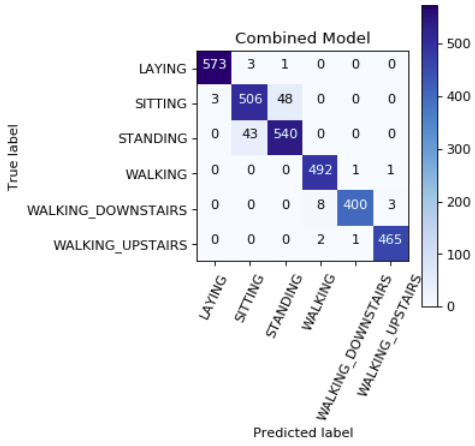

Figure 5: Confusion Matrix for Final model

By this model we combine the prediction capabilites of both KNN and SVC. Final acuracy is around ∼96.5 %.

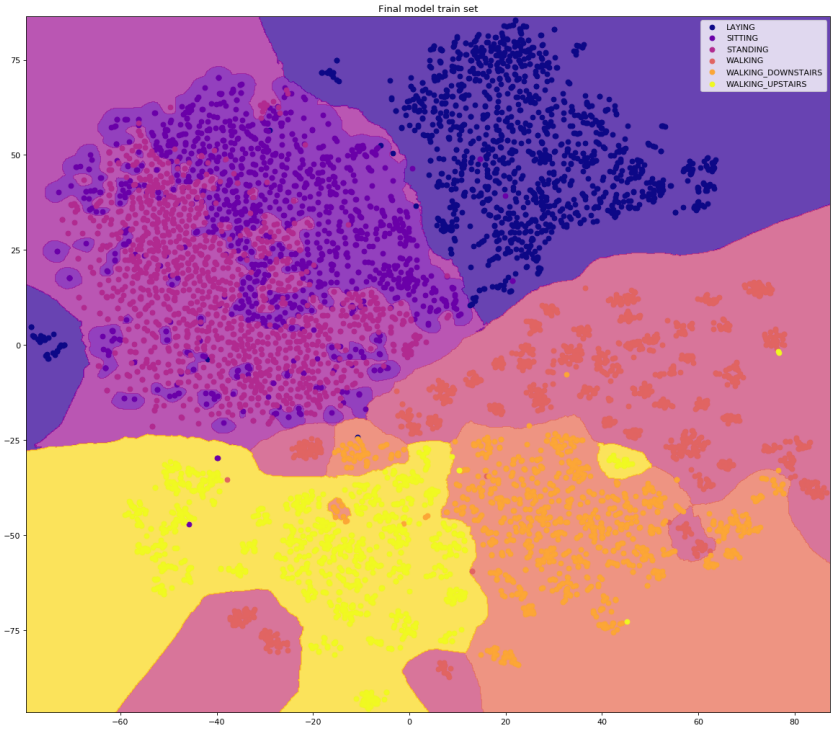

Figure 6: Result on train set

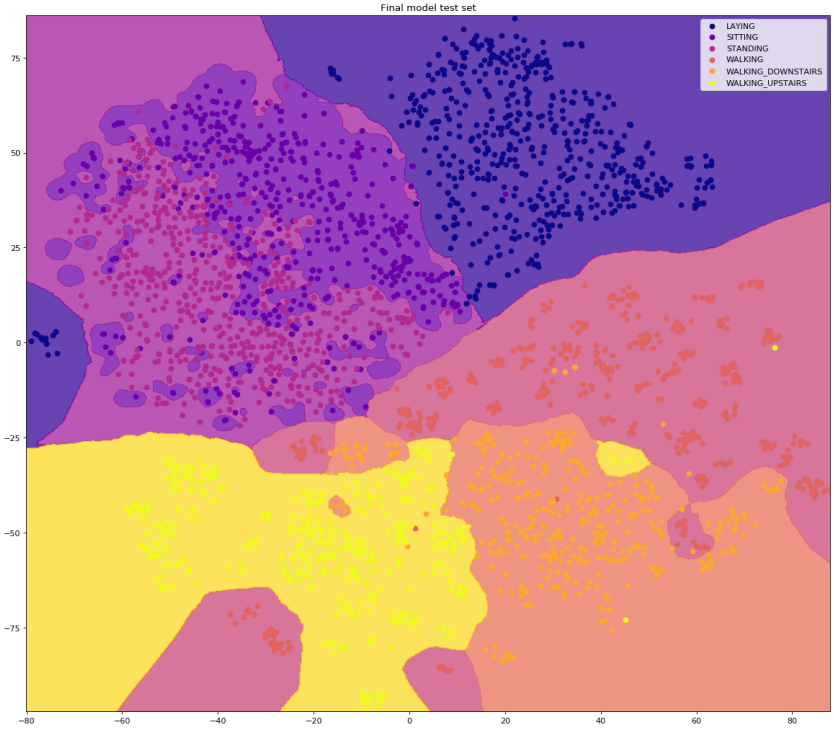

Figure 7: Result on test set

4 Challenges faced

- **Chosing the right model** - I was confused between SVC and KNN. SVC was more accurate where as KNN was more intuitive. At last with Ma'am advice I went with a combined model as metioned above.
- **t-SNE vs PCA and other models** - This was a little tricky. As my data set had 561 dimensions, it was not practical to visual it as such. So I used tsne to gain more insights about the data. Later, as tsne was oreserving most of the data I chosed to train my model in 2 Dimensions.
- **Using Kaggle kernels and notebooks** I learned the power of notebooks and the flexibilty that they offer. This project made me familiar with Kaggle interface and their kernels(which are quite powerfull).

5 Links

Data Set - UCI Machine Learning Repository
Kaggle Kernel - Nikhil
GitHub Repository - nikhilyadv