WQD 7003 Data Analytics

Human Activity Recognition with Neural Network

NG KANG WEI
WQD170068

Contents

Introduction

In the age of big data, advancement in technology churns out all kind of electronic devices. These electronic devices would have internet connectivity and sensors to detect its surroundings. These sensors would collect data about the surroundings and the data would be stored either locally or in a centralized data warehouse. The data collected would be analyzed to provide insights or to predict the future. One of the most ubiquitous technology that had involved in our everyday life, is the smartphone. The smartphone is equipped with a number of sensors that can detect the gyroscopic motion, acceleration, location and other details of the surroundings. The data from the sensors can be collected and used to reveal the activity a person is doing. Activity recognition (AR) enables context aware applications and behavior. AR is also a vital step in the development of mobile health application that need to track a user's activities.

The dataset chosen is the human activity recognition dataset from UCI repository. This dataset is collected from a group of 30 volunteers within the age of 19 to 48 years old. The sensors data is collected from a smartphone, a Samsung Galaxy S II. Every volunteer has to wear the phone on the waist while doing 6 activities. The 6 activities are walking, walking upstairs, walking downstairs, sitting, standing, and laying. The sensors of the smartphone used in this experiment are the accelerometer and the gyroscope. The collected data is the 3-axial linear acceleration and 3-axial angular velocity at the rate of 50Hz.

Based on the data collected by the sensors, the activity done by the subject would be known. The experiment is limited to 6 activities or labels which make this a classification problem. An appropriate machine learning algorithm would be selected to predict the activity of a person based on the collected data.

Research Goal & Objective

The objectives of this research are to

1. Find an appropriate algorithm to predict the activity of a person based on the sensors data collected.
2. The data collected by the researchers has 561 features or columns. Applying all the features into an algorithm would introduce complexity or inaccuracy into the result. Therefore, a number of important features out of the 561 features would be selected.
3. Understand more about the implementation and application of machine learning algorithms on real dataset.

Related works

Human Activity Recognition on Smartphones Using a Multiclass Hardware Friendly Support Vector Machine

The dataset used in this study is collected by a group of researchers, the researchers conducted their own researches on human activity recognition (HAR) with the data. In their research, hardware friendly support vector machine (HF-SVM) is applied to classify the activity. The method chosen uses fixed point arithmetic in the feed forward phase of the SVM classifier which is not resource hungry. The reason for the researchers for considering the limitation of hardware resources when selecting a classification algorithm is because the classification process needs to be performed within the smartphone. A smartphone application is installed within the smartphone that collect the data from the sensors. The sensors data is collected, preprocessed by applying noise filters and sampled in fixed width sliding windows and the activity is classified with HF-SVM.

Figure 1 Activity Recognition pipeline applied by the researchers

The results of the algorithm, HF-SVM is compared with standard floating point Multiclass SVM. The results obtain with HF-SVM is comparable with the results of standard SVM despite HF-SVM is using 6 bits k-value while SVM is using standard 64 bits k-value. (Anguita, Alessandro, Oneto, Parra, & Reyes-Ortiz, 2012)

The end results obtained by the researchers is impressive. They have obtained results with similar accuracy despite they have limited hardware since they need to perform the classification within the smartphone itself.

However, in this study, it is not intended to perform the classification in the smartphone or in a hardware limited environment. Therefore, the classification algorithm chosen would not be limited in this aspect.

Human Activity and Motion Disorder Recognition: Towards Smarter Interactive Cognitive Environments

The researchers from the research on the HAR with HF-SVM has joined a competition in HAR and compile some proposals of the current research on HAR. A general process of HAR can be illustrated in figure 2.

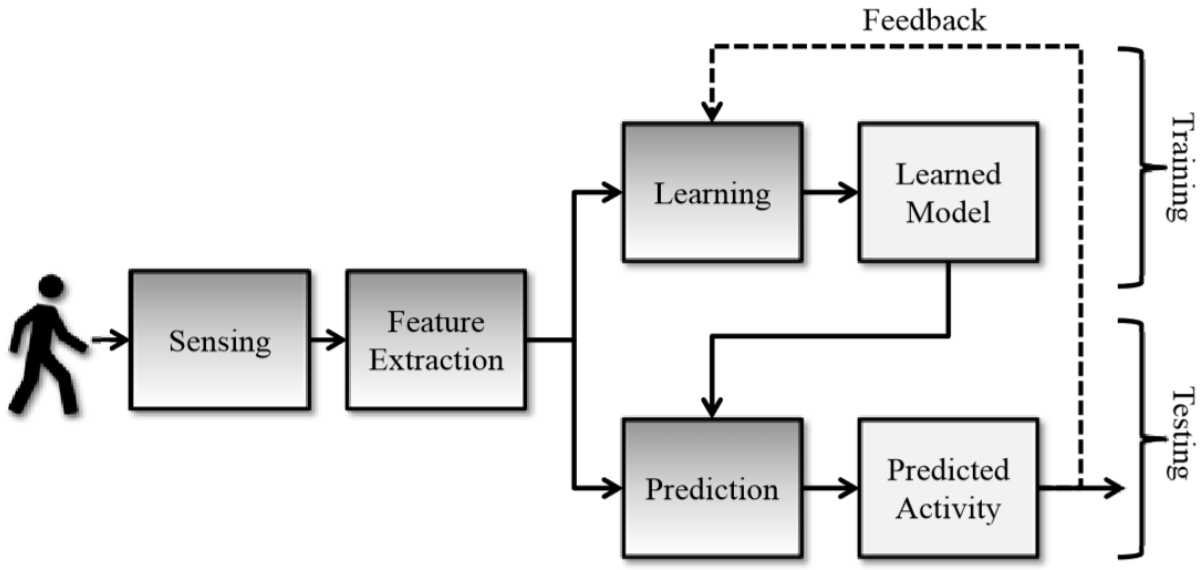

Figure 2 Human Activity Recognition standard process pipeline

This process is suitable for supervises, semi-supervised approaches. Previous work has suggested volunteers to wear 5 accelerometers while performing daily activities. These

devices would be cumbersome to the subjects, as smartphone has accelerometer sensors, it is suggested to use smartphone sensors to replace the body worn sensors. Users just need to keep the accelerometer embedded smartphone in their pocket while doing their daily activities. This is the more convenient and practical method. After solving the sensors and data collection problem, the next problem faced in HAR is the feature extraction. A traditional sense of HAR would divide the sensors signals into time windows with a fixed length. This approach would work fine for the recognition of the body transition from stand to sit or walk to run where short time spans are required, but other more complex activities or in patients with Parkinson's disease a longer time window may be required. The subjects and the usage of the data have to be clarified for different applications, so the result would be accurate. The previous researches on HAR has applied predictive models with geometric approaches such as K-Nearest Neighbors (KNN), Artificial Neural Network (ANN), and Support Vector Machine (SVM) and probabilistic classification methods such as Naïve Bayes and Hidden Markov Models (HMM). The results of all the methods are comparable, there are not clear which algorithm has an edge over another. The performance issue aside, the other aspects such as energy consumptions, memory requirements and computational complexity would the deciding factors. (Reyes-Ortiz, et al., 2013)

The research reviewed all the previous contribution on HAR field and provide a clear picture of how other researchers could proceed or contribute further in HAR. This study is not focus on HAR on a specific medical field, it is on generic HAR, the dataset obtained is for generic HAR.

The Benefits of Personalized Smartphone-Based Activity Recognition Models

In contrary to the methods of HAR proposed above which uses data collected from a panel of representative users, this method proposed that the data is collected personally. If the activity of a person needs to be identified, data must be collected from the person then only apply the classification technique on the data.

The researchers found that with personal training data, the model has higher accuracy than impersonal data. In addition, the amount of training data needed to achieve the accuracy is not as much as impersonal data. (Lockhart & Weiss, 2014)

It is understandable that personalized data has higher accuracy than impersonalized data. Each person has its quirks and habits, there might a slight difference to how each person sits or stands, that might bring some variations into the sensors data. The variations might be the reason for the inaccuracy in the classification sometime. For personalized data, since there is only 1 person involved and the model is applied on the data collected from the said person, the quirks and habits are considered therefore the accuracy would be higher.

However, more effort is needed for personalized data. Data must be collected from person to person and the model must be trained on the personal data for every person. The generic or impersonalized data should achieve a good enough accuracy for almost every person.

Human Activity Recognition using Binary Motion Image and Deep Learning

The main difference of this research is that it does not collect the data from wearable sensors, instead it uses images. The image or picture of a person is obtained and is filtered with Gaussian Mixture Model to obtain the foreground person. The image is now a binary frame, from the binary frame binary motion image (BMI) is calculated and fed into a convoluted neural network (CNN) model for training and testing. The process is illustrated as follows.

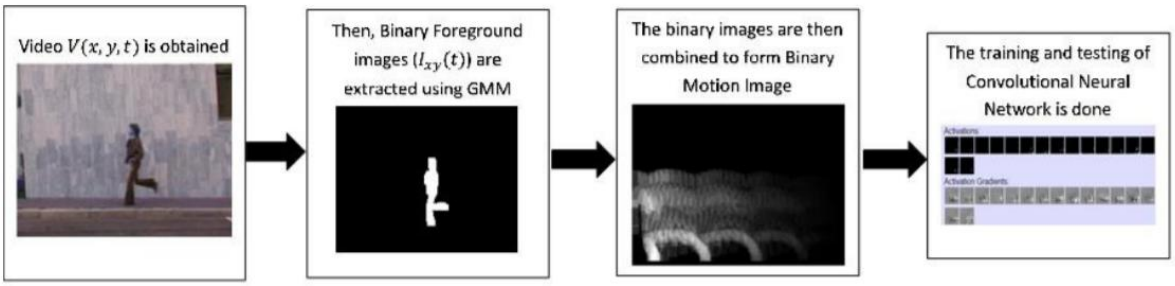

Figure 3 Human Activity Recognition process with Image recognition

The method proposed does not extract complex features from the image as it uses visual templates for recognition. (Dobhal, Shitole, Thomas, & Navada, 2015)

The image method has an advantage over the wearable sensors method as it can be applied in surveillance system to detect the activities of intruders. The research does not compare between the performance of using data collected from wearable sensors data and visual templates, thus there is no clear comparison of result between the 2 methods.

Human Activity Recognition using Neural Networks

This research is using wearable sensors data in HAR but rather than using the smartphone to collect the data, this research uses a smart watch. A Texas Instruments Chronos smart watch has a 3 axes accelerometer incorporated within to identify the acceleration of the body in reference to x, y and z axes. Other than a smart watch, this research also uses a heart rate belt to detect the user's heartbeat. The data collected by the wearable devices are transmitted wirelessly into the storage. The purpose of collecting data with these devices is that a person or a patient's vital signs can be monitored continuously and remotely. This constant monitoring can help early detection in deterioration in health and preventive measures can be taken.

The neural networks used for classification of the data composed of 2 layers of feed forward network with sigmoid activation function on both the hidden and output layers. There are 10 neutrons on the hidden layer and 6 output neurons. The activities recognized by the NN are standing, supine, left lateral recumbent, right lateral recumbent, prone, walking forward, walking backward, running forward, running backward, and sitting. (Oniga & Sütő, 2014)

With neural network the researches achieved 99% accuracy, an almost perfect prediction rate. The use case is different from the dataset collected, this research is more focus on medical application of HAR while the dataset is more generic.

Dataset Preparation

Data Loading

```
setwd('./UCI-HAR-Dataset/')

################ Loading the datasets ###############
# Load the features names
features <- read.delim('./features.txt', header = F, sep = '')
dim(features)
head(features)
# Select the features names from the loaded df
features <- features[,2]

# Load the training dataset
trainData <- read.delim('./train/X_train.txt', header = F, sep = '', dec = '.', col.names = features)
dim(trainData)
head(trainData)

# Load the label for the training data
trainLabel <- read.delim('./train/y_train.txt', header = F, sep = '', col.names = 'activity')
dim(trainLabel)

# Add the label data into a column of the training data
trainData$activity <- trainLabel[,1]

# Activity labels:
# 1-walking
# 2-walking_upstairs
# 3-walking_downstairs
# 4-sitting
# 5-standing
# 6-laying

# Load the test data
testData <- read.delim('./test/X_test.txt', header = F, sep = '', dec = '.', col.names = features)
dim(testData)
head(testData)

# Load the labels for test data
testLabel <- read.delim('./test/y_test.txt', header = F, sep = '', col.names = 'activity')
dim(testLabel)

# Add the test labels to the test data
testData$activity <- testLabel[,1]
```

The HAR dataset is downloaded from UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones. The dataset is separated into training set and testing set. Each of training set and testing set are in text (.txt) files and the response variables, the labels are stored separately from the data. The data is in the form of space separated text file. The data also does not contain the column names or feature names. The features are stored in a separate text file. There are quite some files that need to be loaded and merged to form the complete training and testing dataset.

Since the data is in text files, we cannot use the read.csv function to load the data. The function **read.delim** is used. The separator parameter in the function must be specified to empty, which means one space or more. If the separator parameter is set to a single space, the data would be loaded incorrectly because some of the columns data is not separated by a single space. The code for data loading is shown below.

First, the features name file is loaded. The column of row numbers is removed, and the features names are extracted. Then, the training data is loaded and the features names extracted is passed into the col.names parameter. So that the training data loaded would have the correct column names. After loading the training data, the training label is loaded. The training label is extracted and merged with the training data. Thus, the dataset is complete with the training data and the training labels.

After the training data is loaded correctly, the same steps are taken for the testing data. The data loading stage complete, with both training and testing data loaded correctly.

Sample train data

tBodyAcc.mean...X	tBodyAcc.mean...Y	tBodyAcc.mean...Z	tBodyAcc.std...X	tBodyAcc.std...Y	tBodyAcc.std...Z	tBodyAcc.mad...X	tBodyAcc.mad...Y	tBodyAcc.mad...Z
0.28858451	-0.0202941710	-0.13290514	-0.99527860	-0.983110610	-0.913526450	-0.9951120800	-0.983184570	-0.923527020
0.27841883	-0.0164105680	-0.12352019	-0.99824528	-0.975300220	-0.960321990	-0.9988071900	-0.974914370	-0.957686220
0.27965306	-0.0194671560	-0.11346169	-0.99537956	-0.967187010	-0.978943960	-0.9965199400	-0.963668370	-0.977468590
0.27917394	-0.0262006460	-0.12328257	-0.99609149	-0.983402700	-0.990675100	-0.9970994700	-0.982749840	-0.989302500
0.27662877	-0.0165696550	-0.11536185	-0.99813862	-0.980817270	-0.990481630	-0.9983211300	-0.979671870	-0.990441130
0.27719877	-0.0100978500	-0.10513725	-0.99733496	-0.990486810	-0.995420030	-0.9976274000	-0.990217690	-0.995548900
0.27945388	-0.0196407760	-0.11002215	-0.99692104	-0.967185930	-0.983117830	-0.9970026800	-0.966096710	-0.983116270
0.27743247	-0.0304883030	-0.12536043	-0.99655926	-0.966728430	-0.981585330	-0.9964852500	-0.966313150	-0.982981760
0.27729342	-0.0217506980	-0.12075082	-0.99732847	-0.961245320	-0.983671560	-0.9975957600	-0.957236230	-0.984379280
0.28058569	-0.0099602983	-0.10606516	-0.99480344	-0.972758400	-0.986243870	-0.9954046200	-0.973663220	-0.985641950
0.27688027	-0.0127218050	-0.10343832	-0.99481511	-0.973076920	-0.985357020	-0.9955092700	-0.973947960	-0.985172470
0.27622817	-0.0214413020	-0.10820234	-0.99824595	-0.987213760	-0.992726590	-0.9982512700	-0.985996540	-0.993181880
0.27845700	-0.0204147610	-0.11273172	-0.99913488	-0.984680040	-0.996274240	-0.9990765400	-0.982937020	-0.996410310
0.27717497	-0.0147128020	-0.10675647	-0.99918834	-0.990526380	-0.993365010	-0.9992113500	-0.990687250	-0.992167530
0.29794572	0.0270939080	-0.06166812	-0.98864079	-0.816698600	-0.901906530	-0.9889579500	-0.794280420	-0.888014600
0.27920345	-0.0230201430	-0.12208028	-0.99683904	-0.974848120	-0.983385510	-0.9970938900	-0.973331930	-0.984065350
0.27903836	-0.0148003780	-0.11684896	-0.99694116	-0.981865620	-0.982576530	-0.9972199800	-0.981619640	-0.981336040
0.28013490	-0.0139169510	-0.10637048	-0.99769492	-0.987515670	-0.990407440	-0.9980143200	-0.987954480	-0.992190120
0.27773106	-0.0182107180	-0.10918803	-0.99749074	-0.993221970	-0.996127950	-0.9979030500	-0.992710720	-0.996491820

Sample test data

tBodyAcc.mean...X	tBodyAcc.mean...Y	tBodyAcc.mean...Z	tBodyAcc.std...X	tBodyAcc.std...Y	tBodyAcc.std...Z	tBodyAcc.mad...X	tBodyAcc.mad...Y	tBodyAcc.mad...Z
0.25717778	-0.0232852300	-0.01465376	-0.938404000	-0.920090780	-0.667683310	-0.95250112	-0.9252486700	-0.674302220
0.28602671	-0.0131633590	-0.11908252	-0.975414690	-0.967457900	-0.944958170	-0.98679880	-0.9684013300	-0.945823400
0.27548482	-0.0260504200	-0.11815167	-0.993819040	-0.969925510	-0.962747980	-0.99440345	-0.9707349800	-0.963482670
0.27029822	-0.0326138690	-0.11752018	-0.994742790	-0.973267610	-0.967090680	-0.99527433	-0.9744710100	-0.968897360
0.27483295	-0.0278477880	-0.12952716	-0.993852480	-0.967445480	-0.978294990	-0.99411140	-0.9659525900	-0.977346000
0.27921995	-0.0186203990	-0.11390197	-0.994455230	-0.970416880	-0.965316290	-0.99458514	-0.9694806000	-0.965896860
0.27974586	-0.0182710260	-0.10399988	-0.995819190	-0.976353610	-0.977724680	-0.99599613	-0.9736648500	-0.979252630
0.27460055	-0.0250351300	-0.11683085	-0.995594420	-0.982068920	-0.985262370	-0.99534087	-0.9814849000	-0.984609570
0.27252874	-0.0209540130	-0.11447249	-0.996784130	-0.975906270	-0.986596990	-0.99702932	-0.9737353400	-0.985556480
0.27574570	-0.0103719940	-0.09977589	-0.998373130	-0.986932910	-0.991021900	-0.99866291	-0.9871396500	-0.991084320
0.27859589	-0.0152319280	-0.09890841	-0.998785070	-0.981943060	-0.991378820	-0.99882827	-0.9800149800	-0.991408870
0.27915178	-0.0218794440	-0.10973077	-0.997781030	-0.992951210	-0.985680280	-0.99771026	-0.9926782100	-0.984939520
0.27454383	-0.0231452940	-0.11253960	-0.996204560	-0.991572800	-0.987518280	-0.99652049	-0.9920611400	-0.987127770
0.26906631	-0.0276859670	-0.11017794	-0.996883600	-0.986439500	-0.988479100	-0.99749785	-0.9873889000	-0.989486510
0.27557925	-0.0189364240	-0.09740978	-0.996064690	-0.968225490	-0.980695980	-0.99621791	-0.9646265000	-0.982347130
0.28193083	-0.0048806150	-0.08610614	-0.989075590	-0.959006260	-0.973023880	-0.99378223	-0.9679529100	-0.977847840
0.31107792	-0.0194306250	-0.10186634	-0.936687800	-0.840185850	-0.816826390	-0.94133717	-0.8488326700	-0.812606430
0.26232825	-0.0232570540	-0.12552494	-0.984561110	-0.913379820	-0.912672790	-0.98395147	-0.9072032900	-0.900434710
0.28841575	-0.0034853950	-0.08382786	-0.994570650	-0.978359840	-0.979942620	-0.99533656	-0.9767305900	-0.977797110

Data Preprocessing

```
############ Data preprocessing ######
sapply(trainData, function (x) sum(is.na(x)))
sum(is.na(trainData))

sapply(testData, function(x) sum(is.na(x)))
sum(is.na(testData))

unique(trainData$activity)
unique(testData$activity)

str(trainData, list.len=ncol(trainData))
str(testData, list.len=ncol(testData))

summary(trainData)
summary(testData)

# Change the labels to factor
trainData$activity <- factor(trainData$activity)
testData$activity <- factor(testData$activity)

levels(trainData$activity)
levels(testData$activity)

# Remove some not needed variable to clear up space
rm(features, testLabel, trainLabel)
```

After the data is loaded, the data must check for missing values in the data. Missing values in the data would jeopardize the result of the analysis and should be removed before analysis. From the check, there are no missing values in the dataset. The researchers who uploaded the data must have cleansed the data before. Other than missing values, the structure of the data also must be checked to ensure the format complies with the requirement of the analysis algorithm.

All the columns of the dataset are in numeric format and all the values have been normalized. Numeric values would work fine for all the columns except the response variable column. The response variable column or the activity column contains the labels for the data. It designates the activity of the subject with a number. 1 for walking, 2 for walking upstairs, 3 for walking downstairs, 4 for sitting, 5 for standing and 6 for laying. It would be better to convert the activity column from numeric format to factor. Factor would limit the value of the column to

the 6 values. After converting the column to factor, a sanity check is performed to check the levels of the factor.

The data is cleansed, in the right format, and has no missing values thus proceed to the analysis stage. The last line in the data preprocessing part is to remove some of the not needed variables in the environment to free up space.

Modelling technique

The problem presented in the dataset is a supervised learning and multiclass classification problem. The dataset has 6 response variables or class of activities. The activity of the subject is classified into 1 of the 6 activities based on the sensors data. The main reference article applied HF-SVM on the dataset and achieve an accuracy of 89%. The researchers were limited by hardware resources as they implemented the classification algorithm to run on a smartphone. Through other researches, it is known that other classification algorithms such as k-nearest neighbor (KNN), Hidden Markov Models (HMM), Artificial Neural Network (ANN), Naïve Bayes and support vector machine (SVM), all produce comparable results.

The classification algorithm chosen in this study is the Artificial Neural Network (ANN). Artificial neural network has proven to be efficient in many fields, such as written letter recognition, and image processing. ANN is the building block for deep learning as well. From related work, it is known that ANN is applied in HAR problems as well and achieved good result. However, in the research, the data collection method differs from the dataset used in this study and the purpose or focus of the research is more on medical monitoring purpose.

In this study, ANN would be applied on the HAR dataset from UCI and the result would be compared with the HF-SVM result.

There are 561 features or variables in the dataset. That is quite a number of features and if all the features are fed into a neural network, the computation would be complex. Only a subset of the features should be fed into the neural network. Principal component analysis (PCA) is an algorithm that can solve that problem. PCA is a dimension reduction algorithm. Its purpose is to extract important features from the dataset. A low dimensional dataset would be extracted from the high dimensional dataset with a motive to capture as much information as possible. The important features extracted from the dataset would be fed into the neural network. The workflow of this study is shown below.

Algorithms & Model Validation

Principal Component Analysis (PCA)

```
############## Principal Component Analysis ##############
pcaTrain <- trainData %>% select(-activity)
pcaTest <- testData %>% select(-activity)
pca <- prcomp(pcaTrain, scale = F)

# Variance
pcaVar <- (pca$sdev) ^2

# Proportion of variance
propVar <- pcaVar / sum(pcaVar)
propVar[1:100]

# Plot the scree plot for proportion of variance
plot(propVar, xlab = 'Principal Component', ylab = 'Proportion of variance explained', type = 'b', main = 'Proportion of Variance explained by Principal Components')

# Sanity check with cumulative scree plot
plot(cumsum(propVar), xlab = 'Principal component', ylab = 'Cumulative proportion of variance explained', type = 'b', main = 'Cumulative Proportion of Variance explained by Principal Component')

# From the scree plot, 50 features can explain 90% of the variance

# Create a new dataset from PCA result
tmpTrain <- data.frame(activity = trainData$activity, pca$x)
t <- as.data.frame(predict(pca, newdata = pcaTest))

train2 <- tmpTrain[,1:51]
test2 <- t[, 1:50]
```

As mentioned in the previous section, PCA, a dimension reduction algorithm needs to be carried out to select the most important features in the dataset. This is because there is a total of 561 features in the dataset and using all the features on the neural network would cause the neural network to have many weights, the complexity would be too much.

First, the response variables or the labels are removed from both the train and test dataset. Then, only the train data is passed into the pca function, **prcomp**. Since the dataset is scaled, the scaled parameter of the pca function is set to FALSE. From the result of the PCA function, the variance of the data is calculated and subsequently the proportion of variance. The scree

plot of proportion of variance would be plotted. This would show that how many variables explained how much of the proportion of the variance in the data.

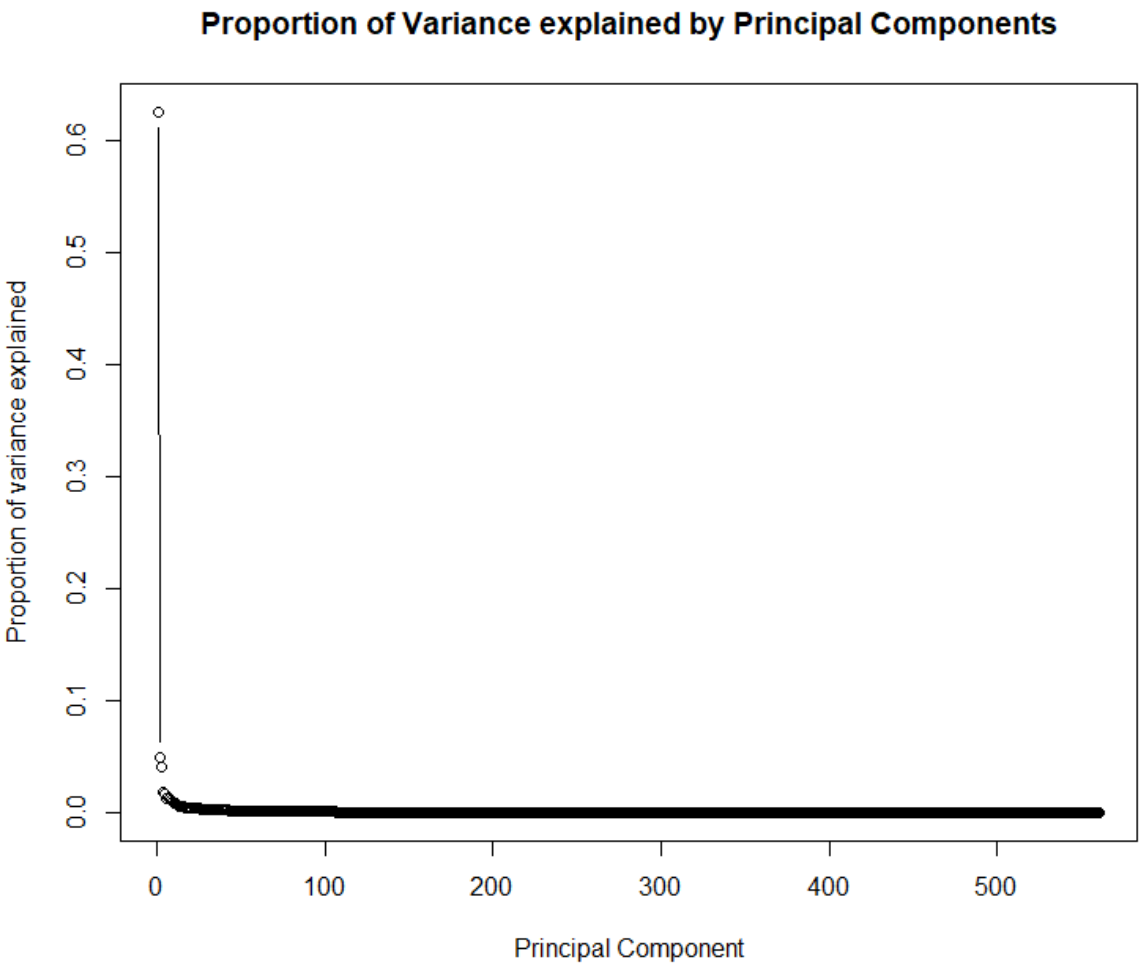

Figure 4 Scree Plot of the proportion of variance

The scree plot shows that about 50 features are needed to explain 99% of the variance of the data. Only 50 features out of the total 561 features are needed. To do a sanity check on this, a cumulative proportion of variance scree plot is plotted as shown in figure 5.

The cumulative proportion of variance scree plot affirm the findings of the previous scree plot. Only 50 features out of the 561 features are needed to explain 99% of the variance in the data.

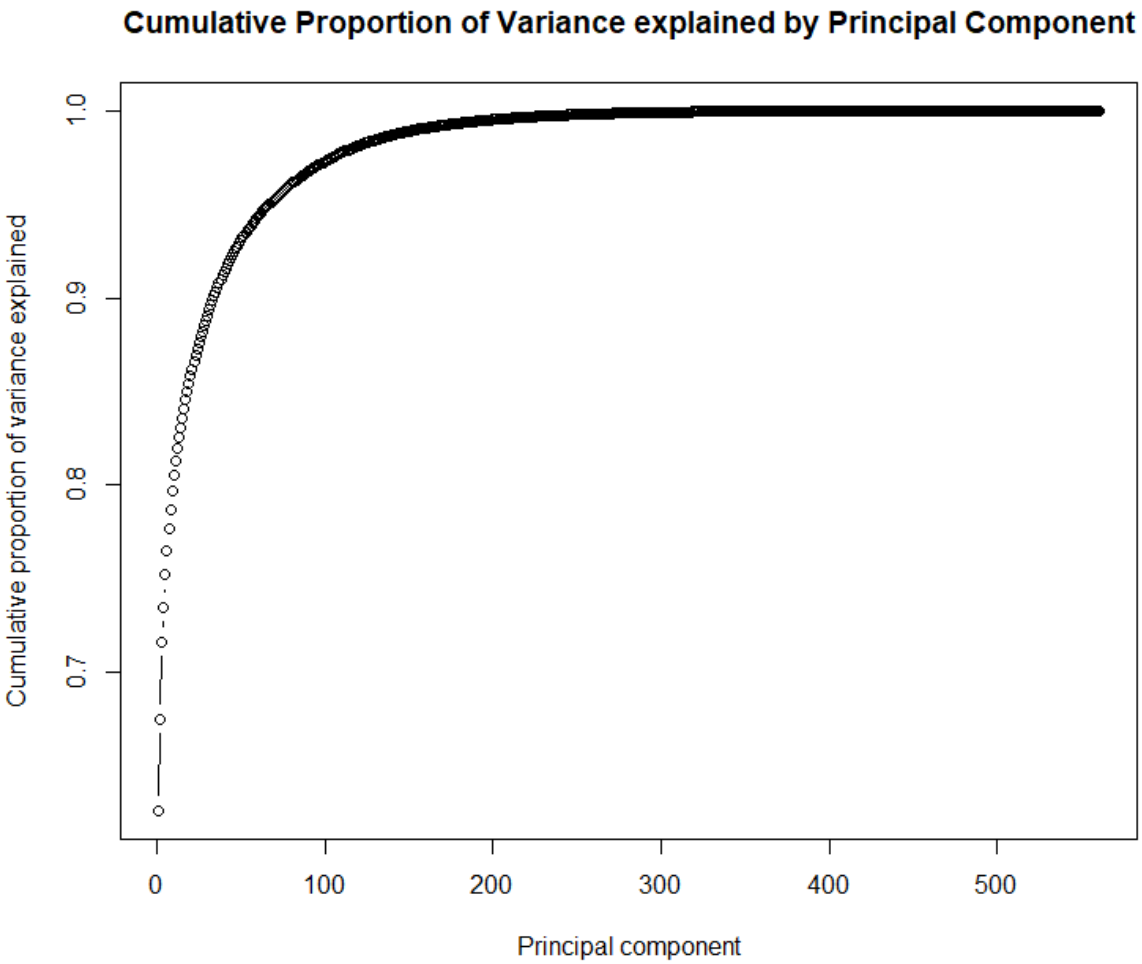

Figure 5 Cumulative Proportion of Variance Scree Plot

From the findings of the PCA, only the first 50 columns of the PCA transformed train data is selected for analysis. The same PCA transformation is performed on the test data and only the first 50 columns are selected.

At this stage, the dimension of both the train and test data are reduced from 561 to 50. The dataset is ready to be fed into neural network.

A side note on PCA, it does not select the 50 features out of 561 features and discard the others. PCA constructs new set of properties based on the combination of the old ones. It transformed all the features into a new space composed by principal component. These new features have no real meaning to us, only algebraic.

Neural Network

```
############### Neural Network after PCA ##################
library(nnet)
library(caret)

# Construct the neural network model with the features from PCA
nn <- nnet(activity ~ ., data = train2, size=10, decay=1.0e-5, maxit=50)

# Apply the neural network model on the testdata
prediction <- predict(nn, test2, type = 'class')

# Check the predicted output
table(prediction)

# Compare the actual output with the predicted
actual <- testData$activity
table(actual)

# Confusion matrix for better comparison
results <- data.frame(actual=actual, prediction=prediction)
t <- table(results)
confusionMatrix(t)
```

At this stage, the principal components obtained from PCA can be fed into a neural network. The neural network library chosen here is **nnet**. The package nnet only provide a single hidden layer neural network, possibly with skip-layer connection.

We would fit all the 50 features into the neural network to predict the outcome of activity. That's the formula part of the neural network. The **decay** parameter is set to a small value, the default is 0. The decay parameter is to help the optimization process and to prevent overfitting in the neural network. The **maxit** parameter is the maximum iteration for the neural network to be trained with the training data. A maximum iteration of 50 seems like a reasonable value. The last parameter to be decided is the **size** or the number of neurons in the hidden layer. The nnet package only provide 1 hidden layer so there only 1 value needed for the size. The value 10 specified in the code above is obtained by trial and error. The neural network is tested with several values of size, and it is found that the accuracy is best with size 10.

After the neural network is trained with the training data. It can be used to predict the outcome or activities in the test data. The prediction and the actual value from the test data are compared in a confusion matrix. The result and the confusion matrix would be discussed in the next section.

Deep Learning

```
############## Deep learning ####################
library(h2o)
library(caret)

# Train an artificial neural network with h2o
h2o.init(nthreads = -1, max_mem_size = '2G')
# Create a clean slate just in case the cluster is already running
h2o.removeAll()
nnModel <-  h2o.deeplearning(y = 'activity',
                    x = setdiff(names(train2), 'activity'),
                    training_frame = as.h2o(train2),
                    activation = 'Rectifier',
                    hidden = c(100,100),
                    distribution = 'multinomial',
                    epochs = 150,
                    train_samples_per_iteration = -2,
                    variable_importances = T)

# Show some statistics of the model
summary(nnModel)
plot(nnModel)

# Use the model to predict the activity on the test data
pred <- h2o.predict(nnModel, newdata = as.h2o(test2))

# Extract the predicted label from the prediction
pred <- as.vector(pred$predict)
table(pred)
actual <- testData$activity
table(actual)

# Construct the confusion matrix
results <- data.frame(actual=actual, prediction=pred)
t <- table(results)
confusionMatrix(t)

# Shut down h2o after completing
h2o.shutdown(prompt = FALSE)
```

Since nnet only provide a single layer neural network, a 2-layer neural network is tested on the dataset to provide a comparison on the results. The package used here is the **h2o** deep learning R package. This package is based on Java and need Java Virtual Machine (JVM) to run.

First, a h2o 1 node server need to be started with **h2o.init** command. The parameters passed in the command, allow the h2o node to utilize all the CPUs and 2 Gb of memory. The **h2o.removeall** command is to remove previous configuration if the node was started, it is to make sure the h2o node is started with a clean slate.

Then, the neural network model is trained with the dataset obtained from PCA. The command to train the neural network is **h2o.deeplearning**. This function has many parameters options, only a few is specified in this case. The **y** parameter specified the labels of the dataset, the **x** parameter is the 50 features of the training data, the **training frame** is the training data in h2o format. The **activation** parameter is set to Rectifier which is the default value. The **distribution** parameter is set to multinomial instead of letting it be determined automatically, the activities classification problem is a multiclass classification problem, so it is explicitly set to multinomial. The **epochs** parameter here is similar to maximum iterations parameter in nnet, it is set to 150. The **train samples per iteration** parameter control the number of training samples per MapReduce iteration, it is set to the default, automatic. The **variable importances** parameter is left to be default as well. The last parameter to be set here is the **hidden** parameter. It specifies the number of neurons in the hidden layer, the default value 200 neurons for 2 hidden layers. 200 neurons for the layers seems excessive and would have high computing cost, so it is set to 100 neurons for 2 hidden layers.

The following plot shows the classification errors decreases as the number of iteration or epoch increases. After epoch 60, the reduction in classification errors is not very significant.

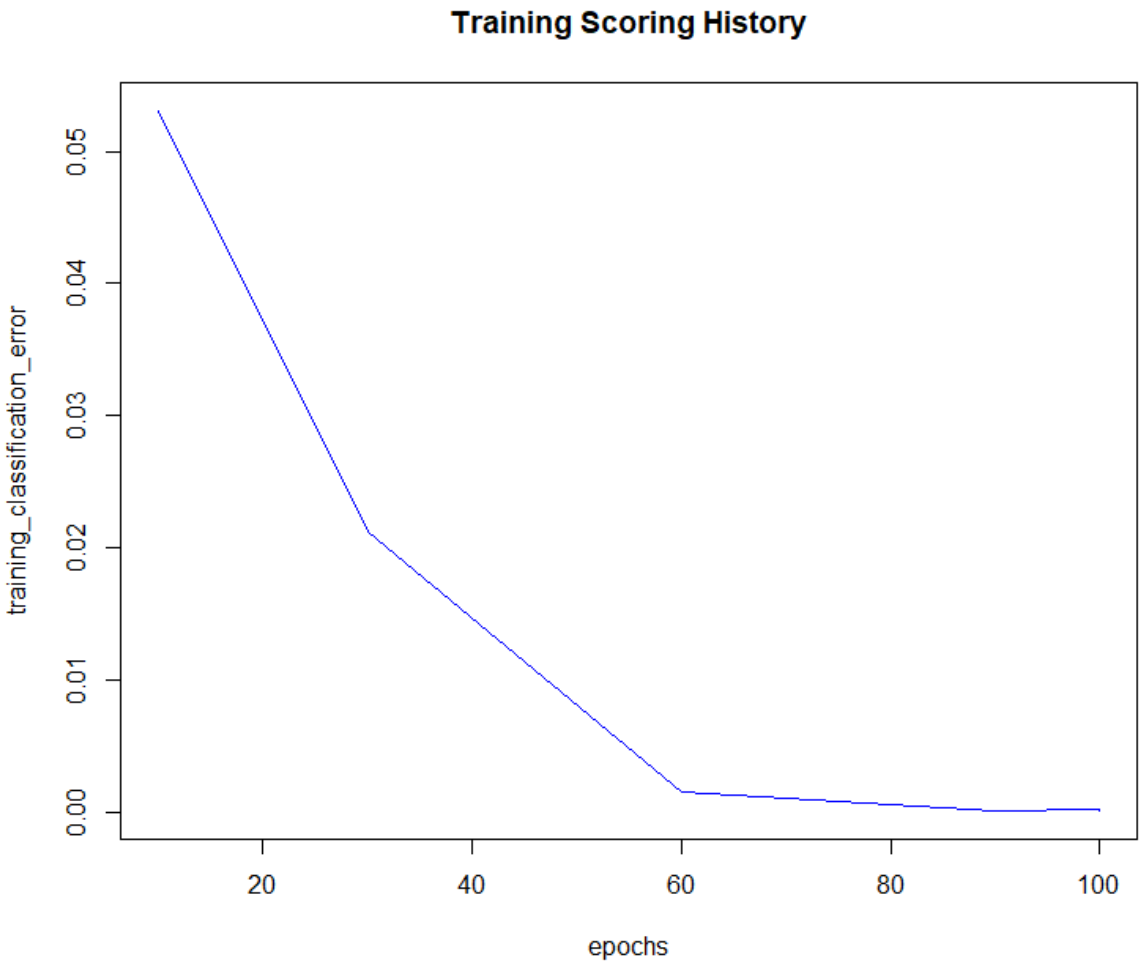

Figure 6 Reduction of training classification errors

Validation Technique

The result of the single layer neural network and 2-layer neural network would be compiled into a confusion matrix. From the confusion matrix, precision and recall for each classification class would be calculated. The performance for each of the activity classified can be compared with the results of the researchers with HF-SVM technique. The results from both the single layer neural network and 2-layer neural network would be compared as well.

Other than precision and recall, the overall accuracy of the classification results is also used to validate or evaluate the performance of the models. The next part of this documentation shows the performance of each algorithm in confusion matrixes and comparison would be made.

Analysis and Recommendation

This section would compare and discuss the prediction results from the previous sections. The result achieved by the researchers using HF-SVM is as follows. (Anguita, Alessandro, Oneto, Parra, & Reyes-Ortiz, 2012).

		Prediction						
		Walking	Walking upstairs	Walking downstairs	Standing	Sitting	Laying	Recall
Actual	Walking	**109**	2	3	0	0	0	95.6
	Walking upstairs	1	**98**	37	0	0	0	72.1
	Walking downstairs	15	14	**114**	0	0	0	79.7
	Standing	0	5	0	**131**	6	0	92.2
	Sitting	0	1	0	3	**108**	0	96.4
	Laying	0	0	0	0	0	**142**	100
	Precision	87.2	81.7	74.0	97.8	94.7	100	89.0

The result of this study with neural network from nnet package:

		Prediction						
		Walking	Walking upstairs	Walking downstairs	Standing	Sitting	Laying	Recall
Actual	Walking	**462**	11	23	0	0	0	93.1
	Walking upstairs	37	**410**	24	0	0	0	87.0
	Walking downstairs	26	41	**353**	0	0	0	84.0
	Standing	0	1	0	**424**	63	3	86.4
	Sitting	3	1	0	52	**476**	0	89.5
	Laying	0	0	0	0	0	**537**	100
	Precision	87.5	88.4	88.3	89.1	88.3	99.4	90.3

The result of this study with 2-layer neural network from h2o package.

		Prediction						
		Walking	Walking upstairs	Walking downstairs	Standing	Sitting	Laying	Recall
Actual	Walking	**474**	9	13	0	0	0	95.6
	Walking upstairs	44	**412**	12	0	2	1	87.5
	Walking downstairs	4	24	**384**	0	8	0	91.4
	Standing	0	2	0	**416**	73	0	84.7
	Sitting	0	0	0	44	**488**	0	91.7
	Laying	0	0	0	1	1	**535**	99.6
	Precision	90.8	92.2	93.9	90.2	85.3	99.8	91.9

The results obtained by a single layer neural network (nnet) and 2-layer neural network (h2o) is better than the result obtained by the researchers with HF-SVM. The researchers obtained an accuracy of 89.0% in the test data, while the single layer and 2-layer neural network obtained an accuracy of 90.3% and 91.9% respectively.

Comparing the precision and recall of HF-SVM and single layer neural network. Single layer neural network did slightly better in the classification of activity walking, walking upstairs and walking downstairs. However, single layer neural network did poorer in the classification of sitting, standing and laying. Similar to HF-SVM, single layer neural network has the highest precision rate at classifying activity laying.

The recall rate of single layer neural network is better in classification of activity walking upstairs and walking downstairs than the recall rate in HF-SVM. In the other activity, categories, single neural network recall rate is slightly lower than that of HF-SVM.

In the case of 2-layer neural network, it performs generally better than both HF-SVM and single layer neural network. In some activity categories, it has slightly lower precision and recall rate than HF-SVM, but it has the highest accuracy among the 3 algorithms.

Even though, 2-layer neural network has the best accuracy, it does not mean that 2-layer neural network is more suitable than HF-SVM to be used within a smartphone app for classifying the activity of the users. The researchers chose HF-SVM for a specific reason, it consumes less memory and need less computing power, consequently, consume less battery. This study on HAR classification is not limited by any hardware restrictions, such as memory, computing power, and battery consumption. A delicate balance between results and the cost need to be achieved.

2-layer neural network performed better than single layer neural network, but the slight increase in accuracy comes at a price. The complexity of the neural network has increased from single layer 10 neurons to 2 layers of 100 neurons to achieve the slight increment in accuracy. The complexity of the algorithm increases with the increment of neurons and layers, the computing power and time needed to train the 2-layer neural network is significantly more than that of a single layer neural network. This steep increase in cost only grant a slight increase in accuracy, precision and recall rate in classification result. A single layer neural network strikes a good balance between performance and resources needed, it achieved a comparable result with less resources.

In the future, ways can be explored to minimize the resources used by the neural network or explore another classification algorithm to achieve comparable results with less resources. There are potentials in the neural network algorithms, many parameters can be tweaked and test its performance again using the current result as benchmark.

Conclusion

This study has explored the neural network in multiclass classification scenario and compare the result obtained with the results shown in the article. The objectives of this study are achieved, neural network is proven to be a viable classification algorithm in the human activity recognition problem. In addition, principal component analysis (PCA) is applied on the 561 features of the dataset. PCA reduced the dimension of the dataset from 561 to 50, which reduced the complexity of the data significantly. Through this study, a better understanding of data analytic process is obtained. This study used real world data, start with data loading and ends with result presentation and discussion. The future work for this study could be to create a neural network algorithm that uses less resources and still produces comparable results.

References

Anguita, D., Alessandro, G., Oneto, L., Parra, X., & Reyes-Ortiz, J. (2012). Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. *Ambient Assisted Living and Home Care. IWAAL 2012* (pp. 216-223). Berlin, Heidelberg: Springer.

Dobhal, T., Shitole, V., Thomas, G., & Navada, G. (2015). Human Activity Recognition using Binary Motion Image and Deep Learning. *Procedia Computer Science* (pp. 178-185). Elsevier B.V.

Lockhart, J. W., & Weiss, G. M. (2014). The Benefits of Personalized Smartphone-based Activity Recognition Models. *SIAM International Conference on Data Mining*, (pp. 614-622).

Oniga, S., & Sütő, J. (2014). Human activity recognition using neural networks. *Proceedings of the 2014 15th International Carpathian Control Conference (ICCC)* (pp. 403-406). IEEE.

Reyes-Ortiz, J., Ghio, A., Anguita, D., Parra, X., Cabestany, J., & Catalia, A. (2013). Human Activity and Motion Disorder Recognition Towards Smarter Interactive Cognitive Environments. *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.* Bruges.

Appendix

```
# Data Analytics
# Name: Ng Kang Wei
# Student ID: WQD170068

library(dplyr)

setwd('./UCI-HAR-Dataset/')

################ Loading the datasets ###############
# Load the features names
features <- read.delim('./features.txt', header = F, sep = '')
dim(features)
head(features)
# Select the features names from the loaded df
features <- features[,2]

# Load the training dataset
trainData <- read.delim('./train/X_train.txt', header = F, sep = '', dec = '.', col.names = features)
dim(trainData)
head(trainData)

# Load the label for the training data
trainLabel <- read.delim('./train/y_train.txt', header = F, sep = '', col.names = 'activity')
dim(trainLabel)

# Add the label data into a column of the training data
trainData$activity <- trainLabel[,1]

# Activity labels:
# 1-walking
# 2-walking_upstairs
# 3-walking_downstairs
# 4-sitting
# 5-standing
# 6-laying

# Load the test data
testData <- read.delim('./test/X_test.txt', header = F, sep = '', dec = '.', col.names = features)
dim(testData)
head(testData)

# Load the labels for test data
testLabel <- read.delim('./test/y_test.txt', header = F, sep = '', col.names = 'activity')
dim(testLabel)

# Add the test labels to the test data
testData$activity <- testLabel[,1]

############ Data preprocessing ######
```

```
sapply(trainData, function (x) sum(is.na(x)))
sum(is.na(trainData))

sapply(testData, function(x) sum(is.na(x)))
sum(is.na(testData))

unique(trainData$activity)
unique(testData$activity)

str(trainData, list.len=ncol(trainData))
str(testData, list.len=ncol(testData))

summary(trainData)
summary(testData)

# Change the labels to factor
trainData$activity <- factor(trainData$activity)
testData$activity <- factor(testData$activity)

levels(trainData$activity)
levels(testData$activity)

# Remove some not needed variable to clear up space
rm(features, testLabel, trainLabel)

############## Principal Component Analysis ##############
pcaTrain <- trainData %>% select(-activity)
pcaTest <- testData %>% select(-activity)
pca <- prcomp(pcaTrain, scale = F)

# Variance
pcaVar <- (pca$sdev) ^2

# Proportion of variance
propVar <- pcaVar / sum(pcaVar)
propVar[1:100]

# Plot the scree plot for proportion of variance
plot(propVar, xlab = 'Principal Component', ylab = 'Proportion of variance explained', type = 'b',
    main = 'Proportion of Variance explained by Principal Components')

# Sanity check with cumulative scree plot
plot(cumsum(propVar), xlab = 'Principal component', ylab = 'Cumulative proportion of variance
explained',
    type = 'b', main = 'Cumulative Proportion of Variance explained by Principal Component')

# From the scree plot, 50 features can explain 90% of the variance

# Create a new dataset from PCA result
tmpTrain <- data.frame(activity = trainData$activity, pca$x)
t <- as.data.frame(predict(pca, newdata = pcaTest))
```

```
train2 <- tmpTrain[,1:51]
test2 <- t[, 1:50]

############### Neural Network after PCA ##################
library(nnet)
library(caret)

# Construct the neural network model with the features from PCA
nn <- nnet(activity ~ ., data = train2, size=10, decay=1.0e-5, maxit=50)

# Apply the neural network model on the testdata
prediction <- predict(nn, test2, type = 'class')

# Check the predicted output
table(prediction)

# Compare the actual output with the predicted
actual <- testData$activity
table(actual)

# Confusion matrix for better comparison
results <- data.frame(actual=actual, prediction=prediction)
t <- table(results)
confusionMatrix(t)

############## Deep learning ####################
library(h2o)
library(caret)

# Train an artificial neural network with h2o
h2o.init(nthreads = -1, max_mem_size = '2G')
# Create a clean slate just in case the cluster is already running
h2o.removeAll()
nnModel <-  h2o.deeplearning(y = 'activity',
                    x = setdiff(names(train2), 'activity'),
                    training_frame = as.h2o(train2),
                    activation = 'Rectifier',
                    hidden = c(100,100),
                    distribution = 'multinomial',
                    epochs = 100,
                    train_samples_per_iteration = -2,
                    variable_importances = T)

# Show some statistics of the model
summary(nnModel)
plot(nnModel)

# Use the model to predict the activity on the test data
pred <- h2o.predict(nnModel, newdata = as.h2o(test2))
```

```
# Extract the predicted label from the prediction
pred <- as.vector(pred$predict)
table(pred)
actual <- testData$activity
table(actual)

# Construct the confusion matrix
results <- data.frame(actual=actual, prediction=pred)
t <- table(results)
confusionMatrix(t)

# Shut down h2o after completing
h2o.shutdown(prompt = FALSE)
```