SVM Gait Identification

Rischan Mafrur

Monday, December 15, 2014

We store the all of result variables from the features extraction to CSV files. Each person has 20 walking data with one data generated by 10 steps. From 20 data, we divide 10 data for training and 10 data for testing. We store training data and testing data in different directory.

This document shows about how to use SVM for gait identification.

The list of files in training and testing directory.

```
setwd("C:/rischan/project/data_testing/csv") #setwd : set the direcotory with param = path
list.files() #list file inside the directory
```

```
## [1] "all_features.csv" "fft_tot.csv"      "tf_tot.csv"
```

```
setwd("C:/rischan/project/data_training/csv")
list.files()
```

```
## [1] "all_features.csv" "fft_tot.csv"      "tf_tot.csv"
```

Data Explanation

In training and testing directories, we have three of files, there are: all_features.csv, it means we combine time domain features and frequency domain features. fft_tot.csv is the data which only contain FFT features and tf_tot.csv the data that only contain time domain features.

All Features Data (* please see the comment inside the code)

```
setwd("C:/rischan/project/data_training/csv")
features <- read.csv(file="all_features.csv",header = TRUE) #read data all features
nrow(features) # Show the number of rows of the data
```

```
## [1] 976
```

```
ncol(features) # show The number of columns
```

```
## [1] 185
```

```
features[1:5,1:5] #show the top data 5 rows, and 5 columns
```

```
##        MeanX        SdX       MaxX        MinX       AbsX
## 1  0.5199714 0.01914113 0.54198208  0.49354279 0.02102035
## 2  0.1471003 0.41283230 0.49354279 -0.77576819 0.19177620
## 3 -0.1757533 0.26739079 0.07076558 -0.64155979 0.19568625
## 4  1.1353718 1.44092694 4.05554551  0.05292161 0.46938241
## 5  1.7914330 0.99649662 4.08005963  0.65756523 0.76751626
```

```
features[972:976,180:185] #show the tail data 5 rows, and 5 columns
```

```
##         FFT156     FFT157     FFT158     FFT159     FFT160   label
## 972 -0.6729655 -0.6816703 -0.6951676 -0.7462649 -0.7605099 rischan
## 973  0.3743154  0.5333078  0.5881150  0.6051548  0.7859553 rischan
## 974  0.1179739  0.3000877  0.2288539  0.3018249  0.2682174 rischan
```

```
## 975 -0.1225885 -0.1662019 -0.1576765 -0.1699950 -0.1457106 rischan
## 976  0.1833701  0.1503869  0.2219942  0.2064059  0.1816273 rischan
```

So the total features when we combine the time domain features and frequency domain features are 184 features (we can see based on number of columns minus one columns is label).

Time domain features data.

Total features from time domain features are 24 (we can see the number of columns minus one (label)).

The lists of features are : Mean, Sd, Max, Min, Abs. Rms (6 features) and we have 4 signals (X,Y,Z, and M). So 6X4, total is 24.

```
setwd("C:/rischan/project/data_training/csv")
features <- read.csv("tf_tot.csv",header = TRUE)
nrow(features) # Show the number of rows of the data
```

```
## [1] 976
```

```
ncol(features) # show The number of columns
```

```
## [1] 25
```

```
features[1:5,1:5] #show the top data 5 rows, and 5 columns
```

```
##        MeanX        SdX       MaxX        MinX       AbsX
## 1  0.5199714 0.01914113 0.54198208  0.49354279 0.02102035
## 2  0.1471003 0.41283230 0.49354279 -0.77576819 0.19177620
## 3 -0.1757533 0.26739079 0.07076558 -0.64155979 0.19568625
## 4  1.1353718 1.44092694 4.05554551  0.05292161 0.46938241
## 5  1.7914330 0.99649662 4.08005963  0.65756523 0.76751626
```

```
features[972:976,10:25] #show the tail data 5 rows, and 5 columns
```

```
##           MinY       AbsY      RmsY      MeanZ        SdZ      MaxZ
## 972 -13.985343 2.59310691 10.430730 -0.8978421 2.31078244 4.4411478
## 973 -12.331570 2.22233020 10.418166  2.0334346 1.48539393 4.4411478
## 974 -11.133841 0.50259782  9.841583  1.1875031 0.61681853 2.2462232
## 975  -9.787127 0.10427519  9.641292  0.6412129 0.18474260 0.9864541
## 976  -9.741904 0.05802428  9.668492  0.5688304 0.07860087 0.7335844
##           MinZ       AbsZ      RmsZ     MeanM        SdM      MaxM
## 972 -3.9717732 2.51468870 2.4373063 10.653197 2.35873182 14.160932
## 973  0.3525772 1.87783118 2.4922808 11.150535 1.41427995 13.083382
## 974  0.6203905 0.33334972 1.3238513 10.051542 0.89412045 11.729947
## 975  0.4822211 0.09435143 0.6644484  9.821273 0.05894976  9.919211
## 976  0.4881956 0.06550816 0.5738213  9.811353 0.09144450  9.967243
##         MinM       AbsM      RmsM   label
## 972 6.917337 2.70186980 10.901387 rischan
## 973 9.061089 2.04300449 11.234632 rischan
## 974 9.189513 0.61935042 10.087269 rischan
## 975 9.742915 0.05113238  9.821431 rischan
## 976 9.665823 0.09898206  9.811747 rischan
```

Frequecy Domain Features.

Similar with previous, we can see the number of features from frequency domain features based on ncol (number of columns -1). The total is 160 features. Its came from 40 FFT coefficients from each signals. We have 4 signals (X,Y,Z,and M).

```r
setwd("C:/rischan/project/data_training/csv")
features <- read.csv("fft_tot.csv",header = TRUE)
nrow(features) # Show the number of rows of the data
```

```
## [1] 976
```

```r
ncol(features) # show The number of columns
```

```
## [1] 161
```

```r
features[1:5,1:5] #show the top data 5 rows, and 5 columns
```

```
##         FFT1       FFT2      FFT3      FFT4      FFT5
## 1  140.58877  100.39239 -69.34285  28.27904 15.716924
## 2  153.58917  -43.09464  18.50395  10.60047  3.333590
## 3 -137.96382   36.98358 -12.48109 124.21126 28.064308
## 4   18.95411 -154.32732 -31.62407 -25.36981 22.217942
## 5  162.78629   52.24382 -23.53280 -51.82805 -8.870268
```

```r
features[972:976,157:161] #show the tail data 5 rows, and 5 columns
```

```
##         FFT157     FFT158     FFT159     FFT160   label
## 972 -0.6816703 -0.6951676 -0.7462649 -0.7605099 rischan
## 973  0.5333078  0.5881150  0.6051548  0.7859553 rischan
## 974  0.3000877  0.2288539  0.3018249  0.2682174 rischan
## 975 -0.1662019 -0.1576765 -0.1699950 -0.1457106 rischan
## 976  0.1503869  0.2219942  0.2064059  0.1816273 rischan
```

SVM for Gait Identification

To use SVM in R, we can use one of SVM library in R. Load the library:

```
#install.packages("e1071")
library("e1071")
```

If you get the problem when you load the library because of your R environment doesnt have this library, just install it using command `install.packages("e1071")`.

In this document we apply SVM three times, there are :

- SVM with the all features (time domain features+frequency domain features)
- SVM only using Time domain features
- SVM only using Frequency domain features

Svm Classification for All Features

```
setwd("C:/rischan/project/data_training/csv")
#SVM model classifier
feature_data <- read.csv("all_features.csv",header = TRUE)
#head(feature_data)

#Divide between the data and label
x <- subset(feature_data, select=-label) #only contain data
y <- feature_data$label #only label

#creating training model
svm_model <- svm(x,y,kernel="radial",cost=1, gamma =0.5)
summary(svm_model) #summary of model
```

```
##
## Call:
## svm.default(x = x, y = y, kernel = "radial", gamma = 0.5, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  963
##
##  ( 241 268 250 204 )
##
##
## Number of Classes:  4
##
## Levels:
##  agung alvin gde rischan
```

```
pred <- predict(svm_model,x)
system.time(pred <- predict(svm_model,x))
```

```
##    user  system elapsed
##    0.39    0.00    0.39
```

```
#Testing of Model Performances
xtab <- table(pred,y)
```

```
#Compute Confusion matrix
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y
## pred      agung alvin gde rischan
##   agung     257     0   0       2
##   alvin       0   238   0       0
##   gde         0     2 270      12
##   rischan     0     1   4     190
##
## Overall Statistics
##
##                Accuracy : 0.9785
##                  95% CI : (0.9673, 0.9866)
##     No Information Rate : 0.2807
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9712
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                1.0000       0.9876     0.9854          0.9314
## Specificity                0.9972       1.0000     0.9801          0.9935
## Pos Pred Value             0.9923       1.0000     0.9507          0.9744
## Neg Pred Value             1.0000       0.9959     0.9942          0.9821
## Prevalence                 0.2633       0.2469     0.2807          0.2090
## Detection Rate             0.2633       0.2439     0.2766          0.1947
## Detection Prevalence       0.2654       0.2439     0.2910          0.1998
## Balanced Accuracy          0.9986       0.9938     0.9827          0.9624
```

Based on confusion matrix above, we achieve 097 % accuracy when we testing performance of our model. It means we use the same data for the create model and also for the testing data.

The performance of our model is good enough, so let's try to using real testing data.

```
setwd("C:/rischan/project/data_testing/csv")
test_data <- read.csv("all_features.csv", header = TRUE) #load testing data
#divide the data and label
x1 <- subset(test_data, select=-label)
y1 <- test_data$label

#apply our model to the testing data (new data)
test_pred <- predict(svm_model,x1)
#see the result
xtab <- table(test_pred,y1)
#Compute Confusion matrix
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y1
## test_pred agung alvin gde rischan
##   agung     266   151 185     161
##   alvin       0    53   8      12
##   gde         2    44  82      31
```

```
##   rischan     0     3   8      29
##
## Overall Statistics
##
##                Accuracy : 0.4155
##                  95% CI : (0.3852, 0.4462)
##     No Information Rate : 0.2734
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.2115
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.9925      0.21116    0.28975        0.12446
## Specificity                0.3520      0.97449    0.89761        0.98628
## Pos Pred Value             0.3486      0.72603    0.51572        0.72500
## Neg Pred Value             0.9926      0.79418    0.77055        0.79497
## Prevalence                 0.2589      0.24251    0.27343        0.22512
## Detection Rate             0.2570      0.05121    0.07923        0.02802
## Detection Prevalence       0.7372      0.07053    0.15362        0.03865
## Balanced Accuracy          0.6723      0.59282    0.59368        0.55537
```

SVM Classification Only using Time Domain Features:

```
setwd("C:/rischan/project/data_training/csv")
#SVM model classifier
feature_data <- read.csv("tf_tot.csv",header = TRUE)
#head(feature_data)
x <- subset(feature_data, select=-label)
y <- feature_data$label

#creating training model
svm_model <- svm(x,y,kernel="radial",cost=1, gamma =0.5)
summary(svm_model)

##
## Call:
## svm.default(x = x, y = y, kernel = "radial", gamma = 0.5, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  821
##
##  ( 200 252 204 165 )
##
##
## Number of Classes:  4
##
## Levels:
##  agung alvin gde rischan
```

```
pred <- predict(svm_model,x)
system.time(pred <- predict(svm_model,x))
```

```
##    user  system elapsed
##    0.10    0.00    0.09
```

```
#Testing of Model Performances
xtab <- table(pred,y)
#Compute Confusion matrix
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y
## pred      agung alvin gde rischan
##   agung     249     1   3       6
##   alvin       1   226   7       1
##   gde         2    11 257      29
##   rischan     5     3   7     168
##
## Overall Statistics
##
##                Accuracy : 0.9221
##                  95% CI : (0.9035, 0.9382)
##     No Information Rate : 0.2807
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8955
##  Mcnemar's Test P-Value : 0.01592
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.9689       0.9378     0.9380         0.8235
## Specificity                0.9861       0.9878     0.9402         0.9806
## Pos Pred Value             0.9614       0.9617     0.8595         0.9180
## Neg Pred Value             0.9888       0.9798     0.9749         0.9546
## Prevalence                 0.2633       0.2469     0.2807         0.2090
## Detection Rate             0.2551       0.2316     0.2633         0.1721
## Detection Prevalence       0.2654       0.2408     0.3064         0.1875
## Balanced Accuracy          0.9775       0.9628     0.9391         0.9020
```

Like previous one, after we check the performance of our model and we think that's enough, we can try that model to the real testing data.

```
setwd("C:/rischan/project/data_testing/csv")
test_data <- read.csv("tf_tot.csv", header = TRUE)
x1 <- subset(test_data, select=-label)
y1 <- test_data$label

test_pred <- predict(svm_model,x1)
#see the result.
xtab <- table(test_pred,y1)
#Compute Confusion matrix
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y1
## test_pred agung alvin gde rischan
##   agung     202    14   6      16
```

```
##   alvin       0   195  16      15
##   gde        58    30 244      55
##   rischan     8    12  17     147
##
## Overall Statistics
##
##                Accuracy : 0.7614
##                  95% CI : (0.7342, 0.787)
##     No Information Rate : 0.2734
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6798
##  Mcnemar's Test P-Value : 6.539e-16
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.7537       0.7769     0.8622          0.6309
## Specificity                0.9531       0.9605     0.8098          0.9539
## Pos Pred Value             0.8487       0.8628     0.6305          0.7989
## Neg Pred Value             0.9172       0.9308     0.9398          0.8989
## Prevalence                 0.2589       0.2425     0.2734          0.2251
## Detection Rate             0.1952       0.1884     0.2357          0.1420
## Detection Prevalence       0.2300       0.2184     0.3739          0.1778
## Balanced Accuracy          0.8534       0.8687     0.8360          0.7924
```

SVM Classification Only using FFT features:

```
setwd("C:/rischan/project/data_training/csv")
#SVM model classifier
feature_data <- read.csv("fft_tot.csv",header = TRUE)
#head(feature_data)
x <- subset(feature_data, select=-label)
y <- feature_data$label

#creating training model
svm_model <- svm(x,y,kernel="radial",cost=1, gamma =0.5)
summary(svm_model)

##
## Call:
## svm.default(x = x, y = y, kernel = "radial", gamma = 0.5, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##       gamma:  0.5
##
## Number of Support Vectors:  933
##
##  ( 232 230 267 204 )
##
##
## Number of Classes:  4
##
```

```
## Levels:
##  agung alvin gde rischan
```

```
pred <- predict(svm_model,x)
system.time(pred <- predict(svm_model,x))
```

```
##    user  system elapsed
##    0.33    0.00    0.33
```

```
#Testing of Model Performances
xtab <- table(pred,y)
#Compute Confusion matrix
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y
## pred      agung alvin gde rischan
##   agung     255     0   2       5
##   alvin       1   234   5       2
##   gde         0     6 266      26
##   rischan     1     1   1     171
##
## Overall Statistics
##
##                Accuracy : 0.9488
##                  95% CI : (0.933, 0.9617)
##     No Information Rate : 0.2807
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9312
##  Mcnemar's Test P-Value : 5.482e-05
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.9922       0.9710     0.9708          0.8382
## Specificity                0.9903       0.9891     0.9544          0.9961
## Pos Pred Value             0.9733       0.9669     0.8926          0.9828
## Neg Pred Value             0.9972       0.9905     0.9882          0.9589
## Prevalence                 0.2633       0.2469     0.2807          0.2090
## Detection Rate             0.2613       0.2398     0.2725          0.1752
## Detection Prevalence       0.2684       0.2480     0.3053          0.1783
## Balanced Accuracy          0.9912       0.9800     0.9626          0.9172
```

Applying our model to the real testing data

```
setwd("C:/rischan/project/data_testing/csv")
test_data <- read.csv("fft_tot.csv", header = TRUE)
x1 <- subset(test_data, select=-label)
y1 <- test_data$label

test_pred <- predict(svm_model,x1)
#see the result.
xtab <- table(test_pred,y1)
#Compute Confusion matrix
confusionMatrix(xtab)
```

```
## Confusion Matrix and Statistics
##
##          y1
```

```
## test_pred agung alvin gde rischan
##   agung     261   129 145     152
##   alvin       1    91  10      15
##   gde         0    26 122      41
##   rischan     6     5   6      25
##
## Overall Statistics
##
##                Accuracy : 0.4821
##                  95% CI : (0.4513, 0.5131)
##     No Information Rate : 0.2734
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3017
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: agung Class: alvin Class: gde Class: rischan
## Sensitivity                0.9739      0.36255     0.4311        0.10730
## Specificity                0.4446      0.96684     0.9109        0.97880
## Pos Pred Value             0.3799      0.77778     0.6455        0.59524
## Neg Pred Value             0.9799      0.82571     0.8097        0.79053
## Prevalence                 0.2589      0.24251     0.2734        0.22512
## Detection Rate             0.2522      0.08792     0.1179        0.02415
## Detection Prevalence       0.6638      0.11304     0.1826        0.04058
## Balanced Accuracy          0.7092      0.66469     0.6710        0.54305
```

From this result , when we only use time domain features we can achieve 71 % accuracy using real testing data but not for the frequency domain features. This result shows that the classification result using FFT coefficient has bad performance. We guess that because of interpolation of each gait cycle to get 40 coefficient. We try to interpolate again the gait cycle because we only have small value from each gait cycle so we could not get 40 coefficient without interpolate again. And the main of problem is we use 16 Hz sampling rate when we collecting the data, so every seconds we only have 16 values. Next, If we want to collect the try to use higher sampling rate.