SVM Applied to CWRU Data

Biswajit Sahoo

The 11 classes (including normal data) that we consider are: There are 10 class for this external load (1 hp). The classes are:

- C1 : Ball defect (0.007 inch)
- C2 : Ball defect (0.014 inch)
- C3 : Ball defect (0.021 inch)
- C4 : Ball defect (0.028 inch)
- C5 : Inner race fault (0.007 inch)
- C6 : Inner race fault (0.014 inch)
- C7 : Inner race fault (0.021 inch)
- C8 : Inner race fault (0.028 inch)
- C9 : Outer race fault (0.007 inch, data collected from 6 O'clock position)
- C10 : Outer race fault (0.014 inch, 6 O'clock)
- C11 : Outer race fault (0.021 inch, 6 O'clock)

Codes

```
library(reticulate)
use_condaenv("r-reticulate")
```

First we will consider all 12 classes and apply SVM to separately to time domain and wavelet domain features.

```
library(e1071)
# Change the lines below and use the path to data in your system
data_time = read.csv(paste("./feature_matrix/12k",
                           "/feature_time_12k_1024_load_1.csv",
                           sep = ""), header = T,nrows = 1265)
data_wav_energy = read.csv(paste("./feature_matrix/12k",
                           "/feature_wav_energy8_12k_1024_load_1.csv",
                           sep = ""), header = T, nrows = 1265)
data_wav_entropy = read.csv(paste("./feature_matrix/12k",
                           "/feature_wav_ent8_shan_12k_1024_load_1.csv",
                           sep = ""), header = T, nrows = 1265)

set.seed(1)
index = c(sample(1:115,35),sample(116:230,35), sample(231:345,35),
          sample(346:460,35),sample(461:575,35),sample(576:690,35),
          sample(691:805,35),sample(806:920,35),sample(921:1035,35),
          sample(1036:1150,35),sample(1151:1265,35))

train_time = data_time[-index,]
train_wav_energy = data_wav_energy[-index,]
train_wav_entropy = data_wav_entropy[-index,]

test_time = data_time[index,]
test_wav_energy = data_wav_energy[index,]
test_wav_entropy = data_wav_entropy[index,]
```

```
# Shuffle data
train_time = train_time[sample(nrow(train_time)),]
train_wav_energy = train_wav_energy[sample(nrow(train_wav_energy)),]
train_wav_entropy = train_wav_entropy[sample(nrow(train_wav_entropy)),]

test_time = test_time[sample(nrow(test_time)),]
test_wav_energy = test_wav_energy[sample(nrow(test_wav_energy)),]
test_wav_entropy = test_wav_entropy[sample(nrow(test_wav_entropy)),]
```

We apply cross-validation over a different set of parameters to obtain best set of parameters. This cross-validation is done by the 'tune' command and the parameters considered are the cost and gamma values as mentioned in the codes. Radial basis is used. The command 'svm_tune$best.model' is the best model obtained from cross validation. This model is used in later lines.

```
set.seed(11)
svm_tune_time = tune(svm,train_time[,-dim(train_time)[2]],
                train_time[,dim(train_time)[2]],kernel = 'radial',
                ranges = list(cost = c(50,100,200,300,400,500),
                              gamma = c(0.01,0.05,0.1,0.5,1)))
pred_train_time = predict(svm_tune_time$best.model,
                          train_time[,-dim(train_time)[2]])
pred_test_time = predict(svm_tune_time$best.model,
                         test_time[,-dim(train_time)[2]])
# Confusion matrix
train_confu_time = table(train_time[,dim(train_time)[2]],pred_train_time)
test_confu_time = table(test_time[,dim(train_time)[2]],pred_test_time)
```

Confusion matrix for time domain features

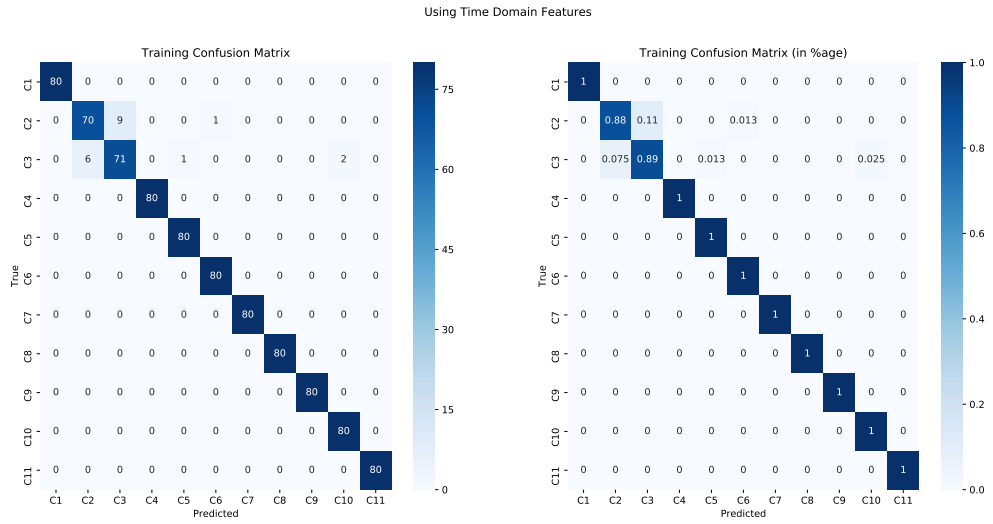
```
import seaborn as sns
import matplotlib.pyplot as plt
fault_type = ['C1','C2','C3','C4','C5','C6','C7',
              'C8','C9','C10','C11']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.train_confu_time, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000020B6D978>
```

```
plt.title('Training Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.subplot(122)
sns.heatmap(r.train_confu_time/80, annot= True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x00000000258E34A8>
```

```
plt.title('Training Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.suptitle('Using Time Domain Features')
plt.show()
```



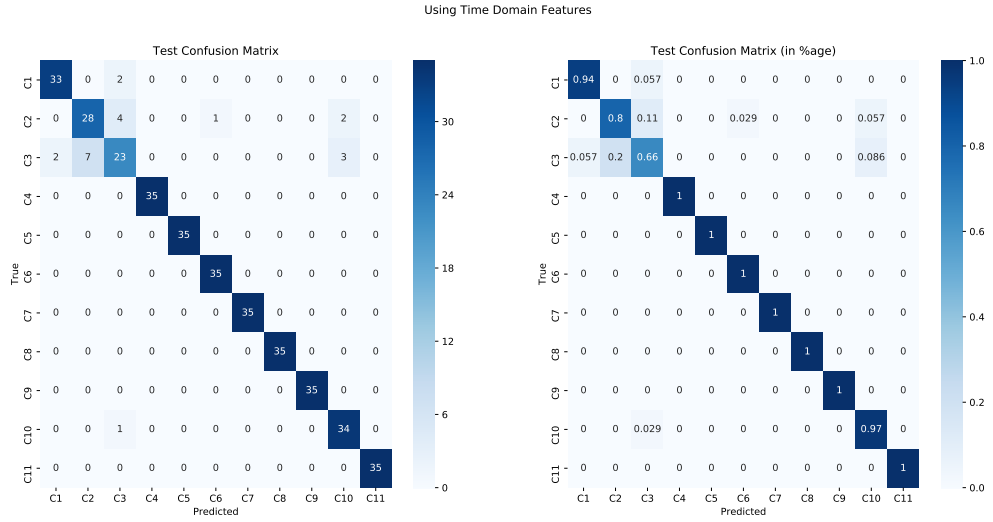

```
plt.figure(2,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu_time, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002603D048>
```

```
plt.title('Test Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.subplot(122)
sns.heatmap(r.test_confu_time/35, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000025F3EDD8>
```

```
plt.title('Test Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.suptitle('Using Time Domain Features')
plt.show()
```




```
overall_test_accuracy_time = sum(diag(test_confu_time))/385
sprintf("Overall Test Accuracy (time features):%.4f",
        overall_test_accuracy_time*100)
```

```
## [1] "Overall Test Accuracy (time features):94.2857"
```

Now for wavelet packet energy and wavelet packet entropy features, we copy the same code with necessary changes.

```
set.seed(12)
svm_tune_energy = tune(svm,train_wav_energy[,-dim(train_wav_energy)[2]],
                train_wav_energy[,dim(train_wav_energy)[2]],
                kernel = 'radial',
                ranges = list(cost = c(50,100,200,300,400,500),
                              gamma = c(0.01,0.05,0.1,0.5,1)))
pred_train_energy = predict(svm_tune_energy$best.model,
                            train_wav_energy[,-dim(train_wav_energy)[2]])
pred_test_energy = predict(svm_tune_energy$best.model,
                           test_wav_energy[,-dim(train_wav_energy)[2]])
# Confusion matrix
train_confu_energy = table(train_wav_energy[,dim(train_wav_energy)[2]],
                           pred_train_energy)
test_confu_energy = table(test_wav_energy[,dim(train_wav_energy)[2]],
                          pred_test_energy)
```

```
fault_type = ['C1','C2','C3','C4','C5','C6','C7',
              'C8','C9','C10','C11']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.train_confu_energy, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

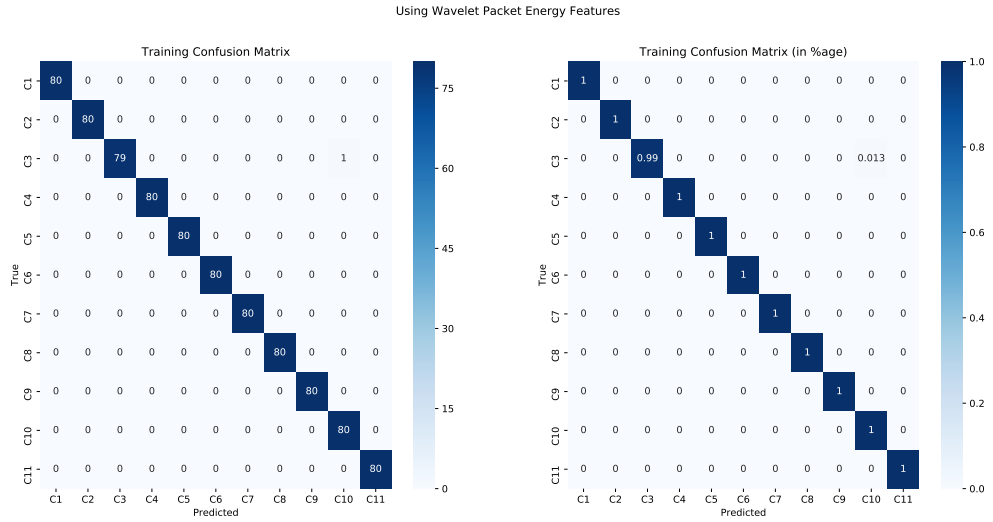
```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000026854160>
```

```
plt.title('Training Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.subplot(122)
```

```
sns.heatmap(r.train_confu_energy/80, annot= True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002258EBBE0>
```

```
plt.title('Training Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.suptitle('Using Wavelet Packet Energy Features')
plt.show()
```



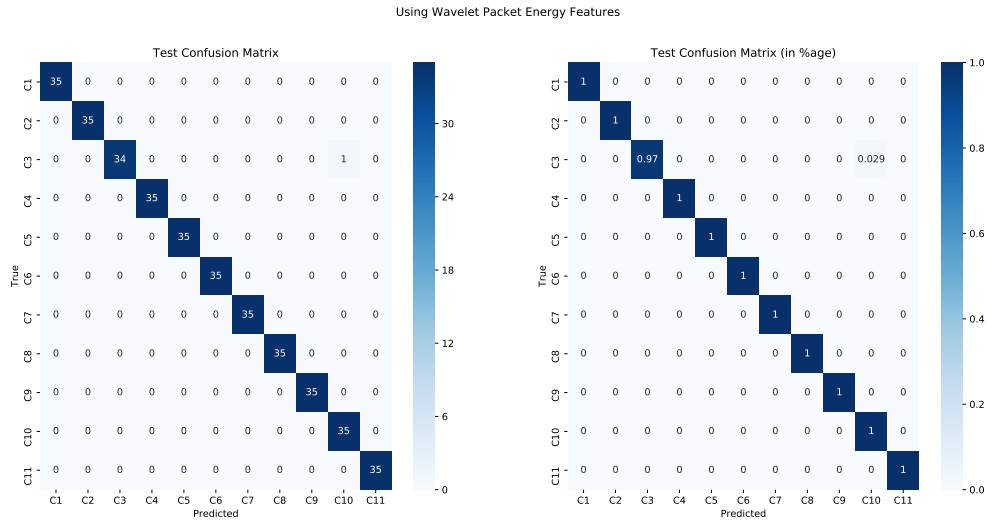

```
plt.figure(2,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu_energy, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002689DC50>
```

```
plt.title('Test Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.subplot(122)
sns.heatmap(r.test_confu_energy/35, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002699E358>
```

```
plt.title('Test Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.suptitle('Using Wavelet Packet Energy Features')
plt.show()
```




```
overall_test_accuracy_energy = sum(diag(test_confu_energy))/385
sprintf("Overall Test Accuracy (wavelet energy features):%.4f",
        overall_test_accuracy_energy*100)
```

```
## [1] "Overall Test Accuracy (wavelet energy features):99.7403"
```

```
set.seed(13)
svm_tune_entropy = tune(svm,train_wav_entropy[,-dim(train_wav_entropy)[2]],
                train_wav_entropy[,dim(train_wav_entropy)[2]],
                kernel = 'radial',
                ranges = list(cost = c(50,100,200,300,400,500),
                              gamma = c(0.01,0.05,0.1,0.5,1)))
pred_train_entropy = predict(svm_tune_entropy$best.model,
                         train_wav_entropy[,-dim(train_wav_entropy)[2]])
pred_test_entropy = predict(svm_tune_entropy$best.model,
                        test_wav_entropy[,-dim(train_wav_entropy)[2]])
# Confusion matrix
train_confu_entropy = table(train_wav_entropy[,dim(train_wav_entropy)[2]],
                          pred_train_entropy)
test_confu_entropy = table(test_wav_entropy[,dim(train_wav_entropy)[2]],
                         pred_test_entropy)
```

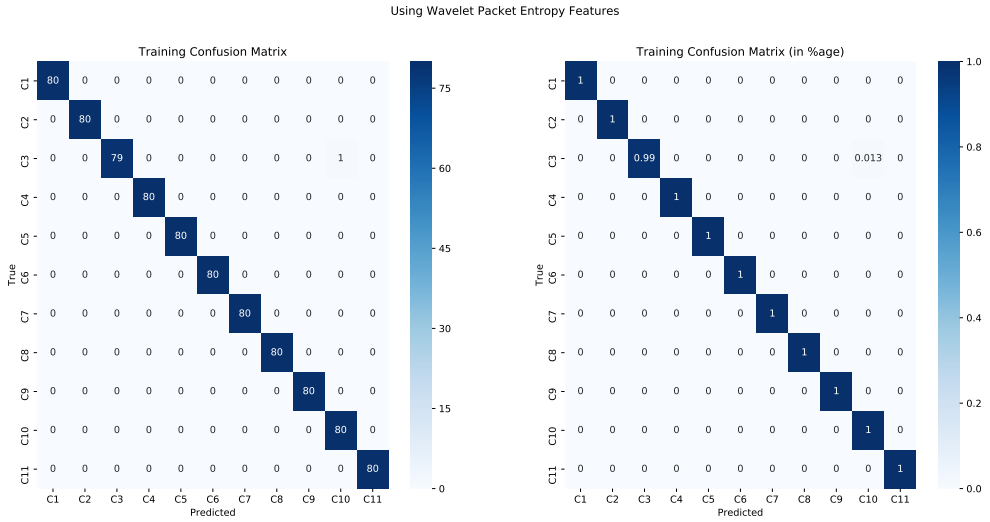
```
fault_type = ['C1','C2','C3','C4','C5','C6','C7',
              'C8','C9','C10','C11']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.train_confu_entropy, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x00000000160889E8>
```

```
plt.title('Training Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.subplot(122)
sns.heatmap(r.train_confu_entropy/80, annot= True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000016081940>
```

```
plt.title('Training Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.suptitle('Using Wavelet Packet Entropy Features')
plt.show()
```



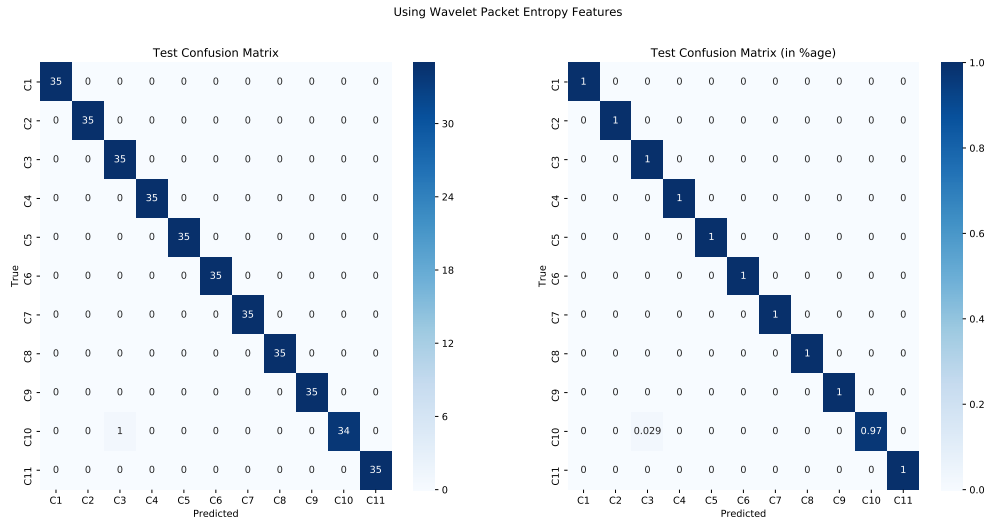

```
plt.figure(2,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu_entropy, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002AE1C128>
```

```
plt.title('Test Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.subplot(122)
sns.heatmap(r.test_confu_entropy/35, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000016101A90>
```

```
plt.title('Test Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.suptitle('Using Wavelet Packet Entropy Features')
plt.show()
```




```
overall_test_accuracy_entropy = sum(diag(test_confu_entropy))/385
sprintf("Overall Test Accuracy (wavelet entropy features):%.4f",
        overall_test_accuracy_entropy*100)
```

```
## [1] "Overall Test Accuracy (wavelet entropy features):99.7403"
```