SVM Applied to CWRU Data (Sampling Frequency 12k)

Biswajit Sahoo

(For other results on CWRU dataset visit this page)

Case Western Reserve University bearing data sets are collected at two sampling frequencies, 48k and 12k. In the previous posts we have used the 48k data. In this post we will use 12k data. The 12k sampling frequency data are of shorter length than 48k data. So we will decrease the segment length from 2048 data points to 1024 data points.

The original data have been collected with 0,1,2, and 3 hp load for each fault type. In this study we have only taken data with 1 hp load.

Important Note: In the CWRU website, sampling frequency of the normal data is not mentioned. Most research paper take it as 48k. Some authors also consider it as being taken at a sampling frequency of 12k. Some other authors just use it without ever mentioning its sampling frequency. But in our application we only need segment of normal data of length 1024. So we will use the normal data segments available at the website without going into the discussion of sampling frequency. Still, to be on the safer side, we will show results including the normal data as a class as well as excluding it.

The 12 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 : Normal
- C10 : Outer race fault (0.007 inch, data collected from 6 O'clock position)
- C11 : Outer race fault (0.014 inch, 6 O'clock)
- C12 : Outer race fault (0.021 inch, 6 O'clock)

When we exclude normal data, we will don't consider "C9" class and study the rest 11 fault classes. We will consider both time domain and wavelet domain features. We will not go into the details here. Refer the link given at the top of the page to get more details.

Download the data from the links below:

- Time domain features 12k
- Wavelet packet energy features 12k
- Wavelet packet entropy (Shannon) features 12k

As 12k data are shorter, for each fault type we collect 115 data segments of 1024 length each. So there are a total of $115 \times 12 = 1380$ data points. Of this we take $35 \times 12 = 420$ data points as test set and rest as training set.

SVM is applied to training set data and best parameters are chosen by cross validation. The best parameters are then applied to test set data to predict final classification result. We have not printed the intermediate results. Readers can easily print those as per requirement. We will use R to implement SVM. To plot confusion matrix, we will use Python.

Codes

```r
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.

```r
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)
data_wav_energy = read.csv(paste("./feature_matrix/12k",
                           "/feature_wav_energy8_12k_1024_load_1.csv",
                           sep = ""), header = T)
data_wav_entropy = read.csv(paste("./feature_matrix/12k",
                           "/feature_wav_ent8_shan_12k_1024_load_1.csv",
                           sep = ""), header = T)

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),sample(1266:1380,35))

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.

```r
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

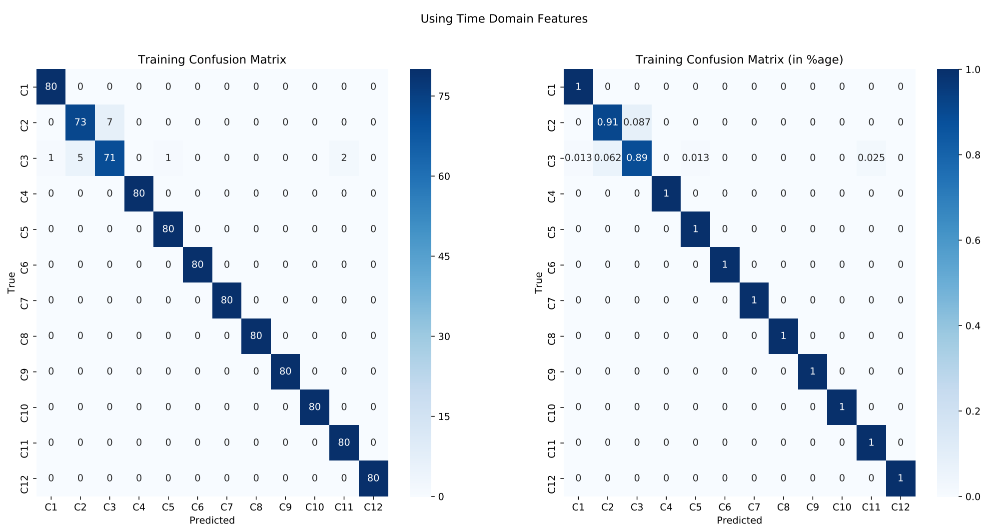
```
import seaborn as sns
import matplotlib.pyplot as plt
fault_type = ['C1','C2','C3','C4','C5','C6','C7',
              'C8','C9','C10','C11','C12']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.train_confu_time, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000023844748>
```

```
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 0x00000000259166D8>
```

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



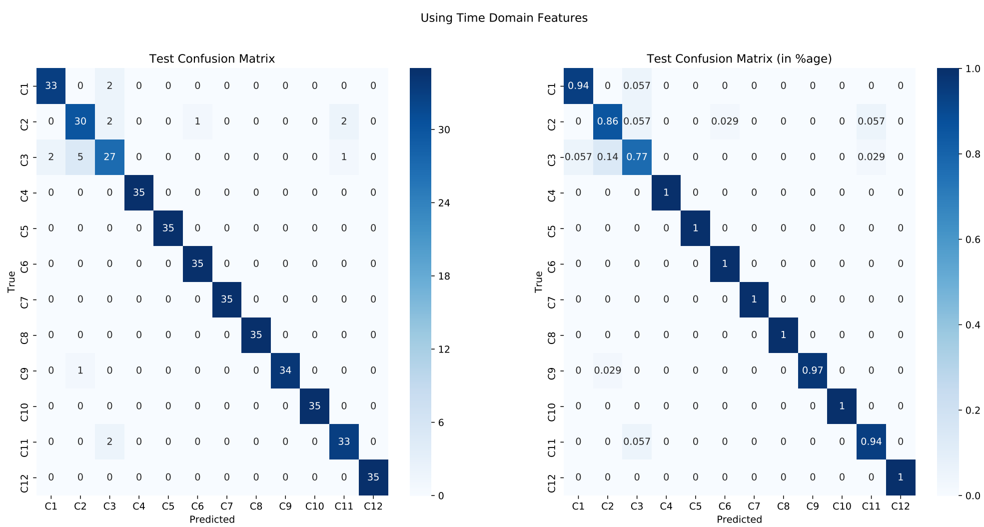

```
plt.figure(2,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu_time, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000025FD0A58>
```

```
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 0x000000002590DDA0>
```

```
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))/420
sprintf("Overall Test Accuracy (time features):%.4f",
        overall_test_accuracy_time*100)
```

```
## [1] "Overall Test Accuracy (time features):95.7143"
```

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)
```

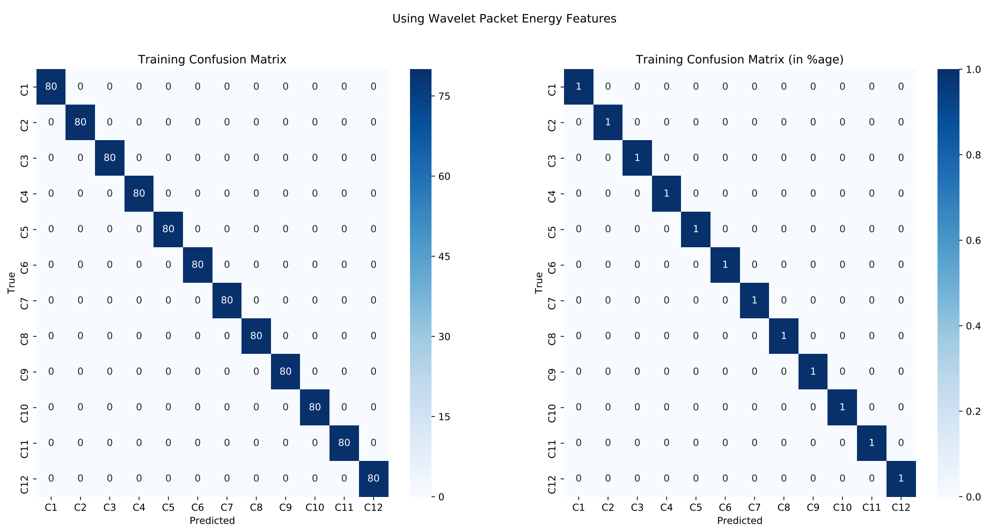
```
fault_type = ['C1','C2','C3','C4','C5','C6','C7',
              'C8','C9','C10','C11','C12']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.train_confu_energy, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000026847470>
```

```
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 0x000000002698D160>
```

```
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")
```

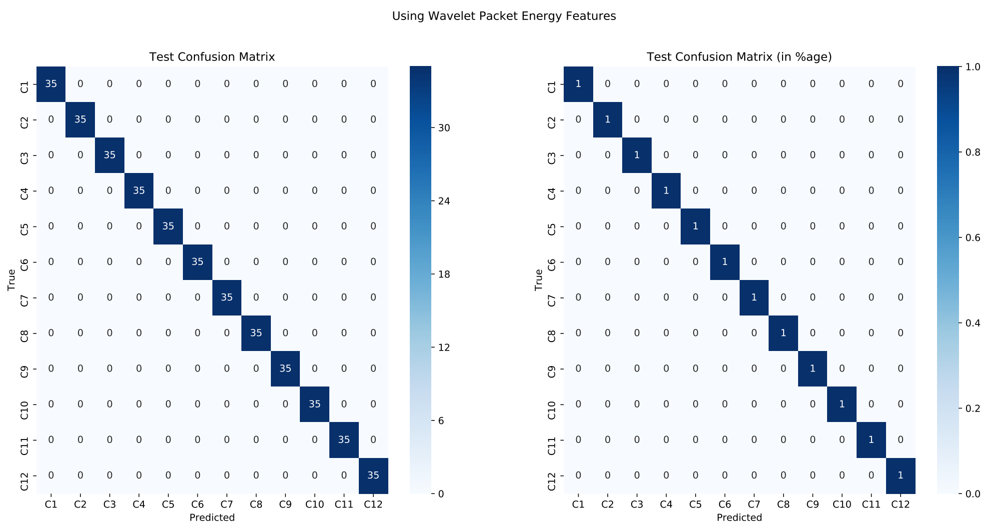
```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000015C7BB38>
```

```
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 0x00000000258833C8>
```

```
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))/420
sprintf("Overall Test Accuracy (wavelet energy features):%.4f",
        overall_test_accuracy_energy*100)
```

```
## [1] "Overall Test Accuracy (wavelet energy features):100.0000"
```

```
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','C12']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
```

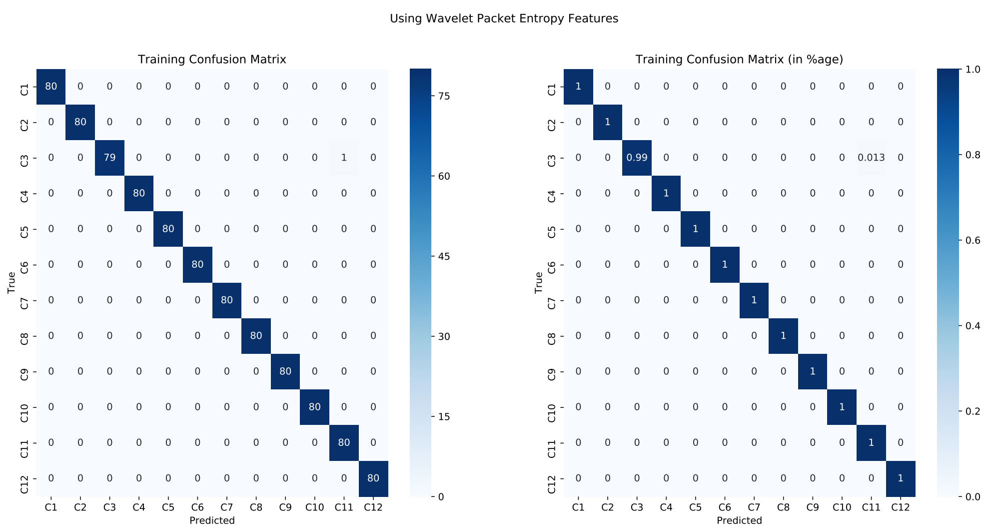
```
sns.heatmap(r.train_confu_entropy, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000015ECB320>
```

```
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 0x0000000015FFBCF8>
```

```
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")
```

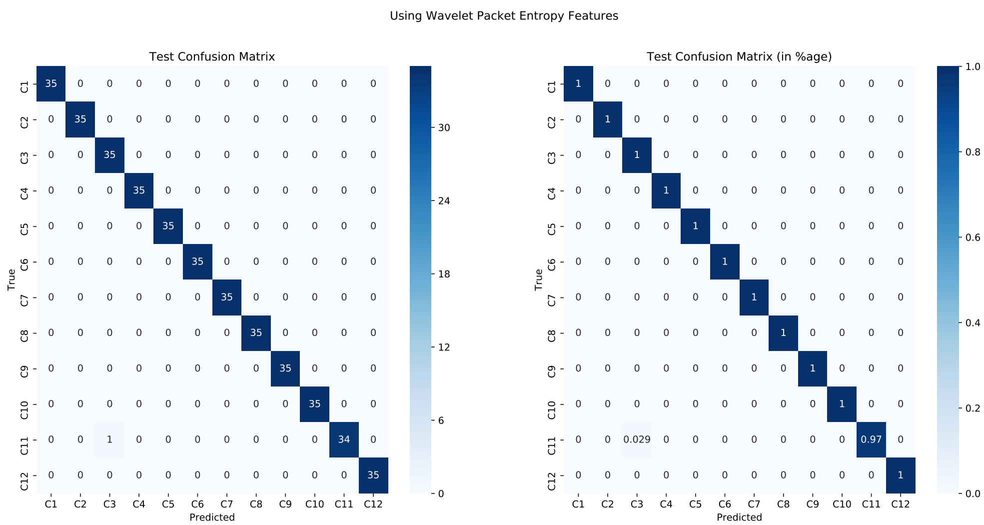
```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000015B70D68>
```

```
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 0x000000002686ECC0>
```

```
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))/420
sprintf("Overall Test Accuracy (wavelet entropy features):%.4f",
        overall_test_accuracy_entropy*100)
```

```
## [1] "Overall Test Accuracy (wavelet entropy features):99.7619"
```

Note that with wavelet packet features accuracy is near perfect. This is also in agreement with our previous results on 48k data.

(To get results without normal class, go to this link)

Last modified: 14^{th} June, 2019