Quadratic Discriminant Analysis for Bearing Fault Classification

Biswajit Sahoo

In a previous post we had discussed LDA. In LDA, a common covariance matrix is assumed for all classes. But in Quadratic Discriminant Analysis (QDA) different covariance matrices are used for different classes. Thus number of parameters in QDA is much larger than LDA and it is prone to high variance. But if data satisfies the underlying assumptions of QDA and the dataset is not to small, QDA gives better results.

We will postpone the details of the method to a later time. For the time being we direct the interested reader to this excellent book.

We will provide codes in R to show that LDA also works well for bearing fault classification. We will use package 'MASS' to implement QDA.

Description of data

Detailed discussion of how to prepare the data and its source can be found in this post. Here we will only mention about different classes of the data. There are 12 classes and data for each class are taken at a load of 1hp. 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)

Important Note: In the CWRU website, sampling frequency for 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. 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.

When we exclude normal data, we won't consider "C9" class and study the rest 11 fault classes. At that time "C09", "C10", and "C11" will correspond to outer race faults of fault depth 0.007, 0.014, and 0.021 inch respectively.

Codes

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

How to get data?

Readers can download the `.csv` file used in this notebook from here. Another convenient way is to download the whole repository and run the downloaded notebooks.

```
library(MASS)
data_wav_energy = read.csv("./data/feature_wav_energy8_12k_1024_load_1.csv",
                           header = T)
# Change the above line to include your folder that contains data
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_data = data_wav_energy[-index,]
test_data = data_wav_energy[index,]

# Shuffle data
train_data = train_data[sample(nrow(train_data)),]
test_data = test_data[sample(nrow(test_data)),]
```

It should be noted that for some of the deterministic techniques, shuffling of data is not required. But some other techniques like deep learning require the data to be shuffled for better training. So as a recipe we always shuffle data whether the method is deterministic or not. This doesn't hurt either for a deterministic technique.

```
qda_fit = qda(fault~., train_data)
pred_train = predict(qda_fit, newdata = train_data)
pred_test = predict(qda_fit, newdata = test_data)
# Confusion matrix
train_confu = table(train_data$fault, pred_train$class)
test_confu = table(test_data$fault, pred_test$class)
```

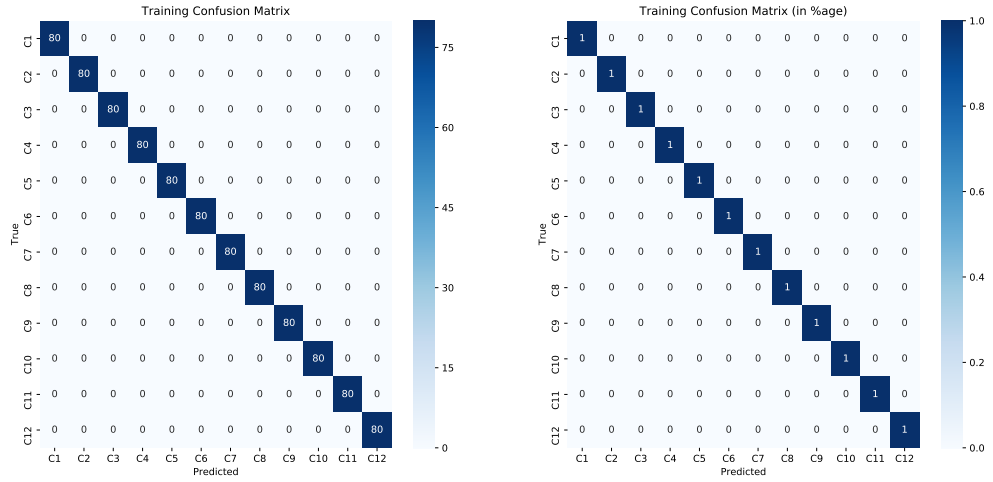
```
import seaborn as sns
import matplotlib.pyplot as plt
fault_type = ['C1','C2','C3','C4','C5','C6','C7','C8','C9','C10','C11','C12']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.train_confu, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000018ED12C8>
```

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

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000029BF4348>
```

```
plt.title('Training Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



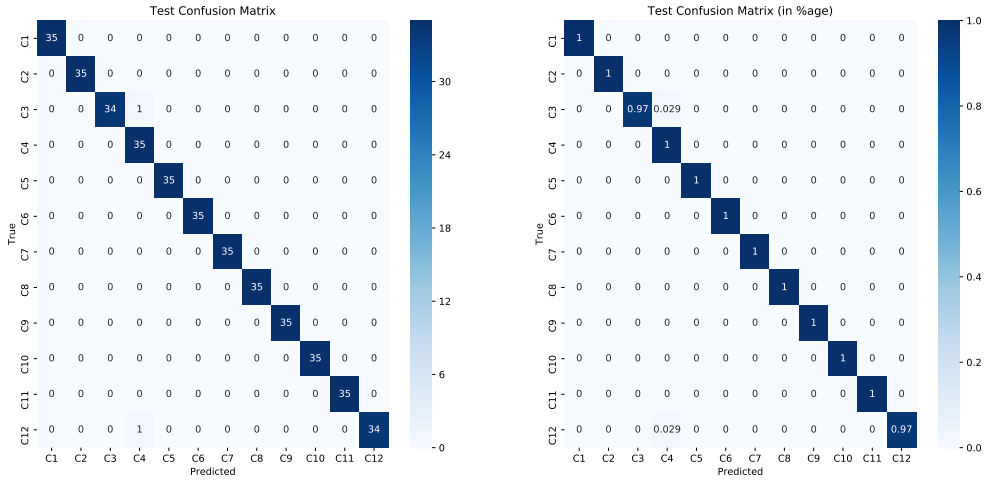

```
plt.figure(2,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002A55C6C8>
```

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

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002A374208>
```

```
plt.title('Test Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```




```
overall_test_accuracy = sum(diag(test_confu))/420
sprintf("Overall Test Accuracy: %.4f", overall_test_accuracy*100)
```

```
## [1] "Overall Test Accuracy: 99.5238"
```

We will also show the results excluding the normal data. The results are as below.

```
data_without_normal = read.csv("./data/feature_wav_energy8_12k_1024_load_1.csv",
                               header = T, nrows = 1265)
# Change the above line to include your folder that contains data
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_new = data_without_normal[-index,]
test_new = data_without_normal[index,]

# Shuffle data
train_data_new = train_new[sample(nrow(train_new)),]
test_data_new = test_new[sample(nrow(test_new)),]
```

```
qda_fit_new = qda(fault~., train_data_new)
pred_train_new = predict(qda_fit_new, newdata = train_data_new)
pred_test_new = predict(qda_fit_new, newdata = test_data_new)
# Confusion matrix
train_confu_new = table(train_data_new$fault, pred_train_new$class)
test_confu_new = table(test_data_new$fault, pred_test_new$class)
```

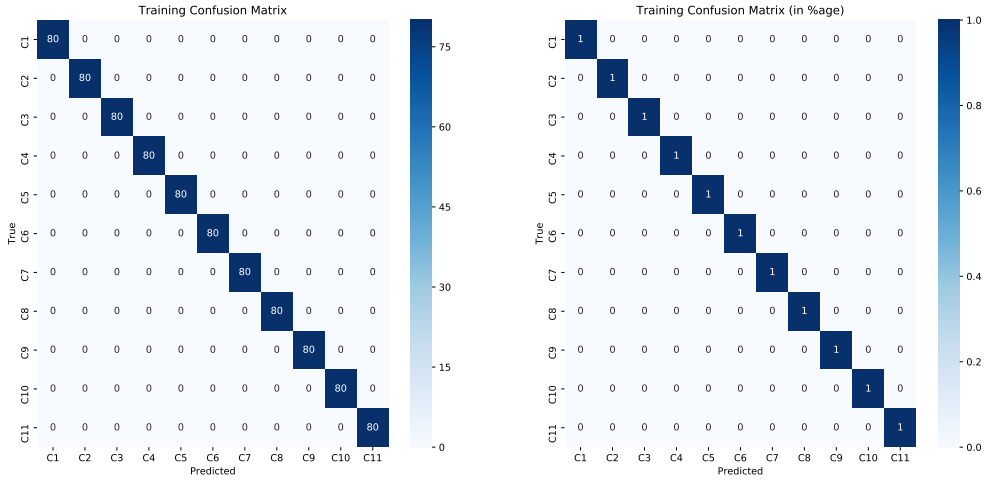
```
import seaborn as sns
import matplotlib.pyplot as plt
fault_type = ['C1','C2','C3','C4','C5','C6','C7','C8','C9','C10','C11']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.train_confu_new, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002B6B9488>
```

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

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000029C35848>
```

```
plt.title('Training Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```



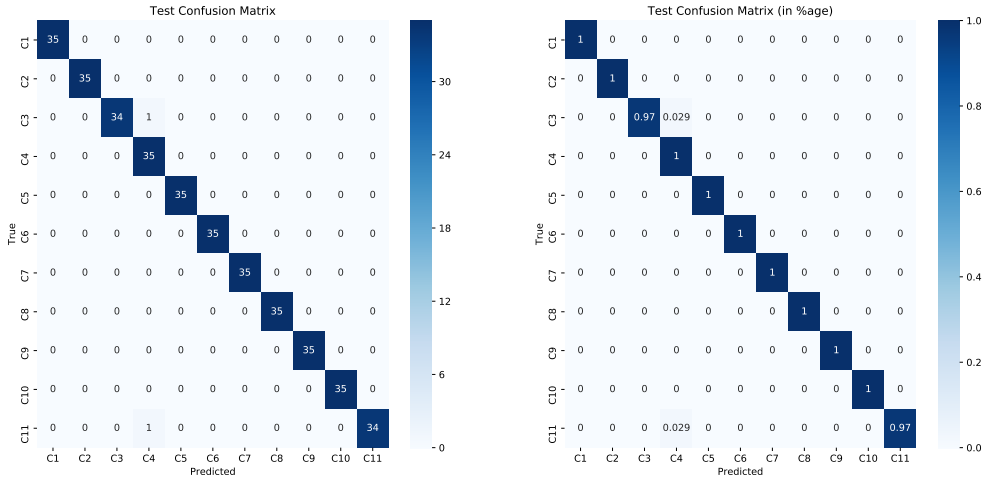

```
plt.figure(2,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu_new, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002A2B0B08>
```

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

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000029C86F08>
```

```
plt.title('Test Confusion Matrix (in %age)')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.show()
```




```
overall_test_accuracy_new = sum(diag(test_confu_new))/385
sprintf("New overall Test Accuracy: %.4f", overall_test_accuracy_new*100)
```

```
## [1] "New overall Test Accuracy: 99.4805"
```

To see results of other techniques applied to public condition monitoring datasets, visit this page.

```
sessionInfo()
```

```
## R version 3.6.2 (2019-12-12)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] MASS_7.3-51.4   reticulate_1.14
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.3      digest_0.6.23   rappdirs_0.3.1  jsonlite_1.6.1
##  [5] magrittr_1.5    evaluate_0.14   rlang_0.4.4     stringi_1.4.5
##  [9] rmarkdown_2.1   tools_3.6.2     stringr_1.4.0   xfun_0.12
## [13] yaml_2.2.0      compiler_3.6.2  htmltools_0.4.0 knitr_1.27
```

Last updated: 14^{th} February, 2020