Linear Discriminant Analysis for Bearing Fault Classification

Biswajit Sahoo

Linear Discriminant Analysis (LDA) is a classification technique that assumes a Gaussian distribution over variables for each class. Different means are chosen for different classes but same covariance matrix is chosen for all classes. Taking same covariance matrix for all classes is indeed a restricting assumption but it works well when this assumption is approximately satisfied and number of data points is less. Once Gaussians are fit for each class, posterior probability of a data point belonging to a particular class is determined by applying Bayes' rule. The parameters of the distribution are learnt form data.

We will add a detailed theory of LDA to this post at 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 LDA.

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")
```

First download the data from here. Save the data in a folder and read it from that folder.

```r
library(MASS)
data_wav_energy = read.csv("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.

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

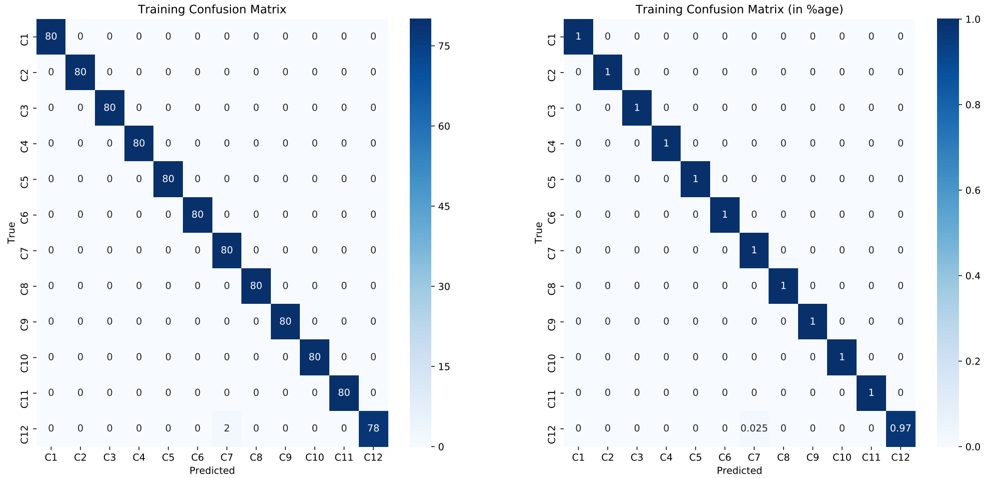
```python
import seaborn as sns
import matplotlib.pyplot as plt
fault_type = ['C1','C2','C3','C4','C5','C6','C7','C8','C9','C10','C11','C12']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.train_confu, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")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

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000023E285C0>
```

```python
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 0x0000000026579390>
```

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



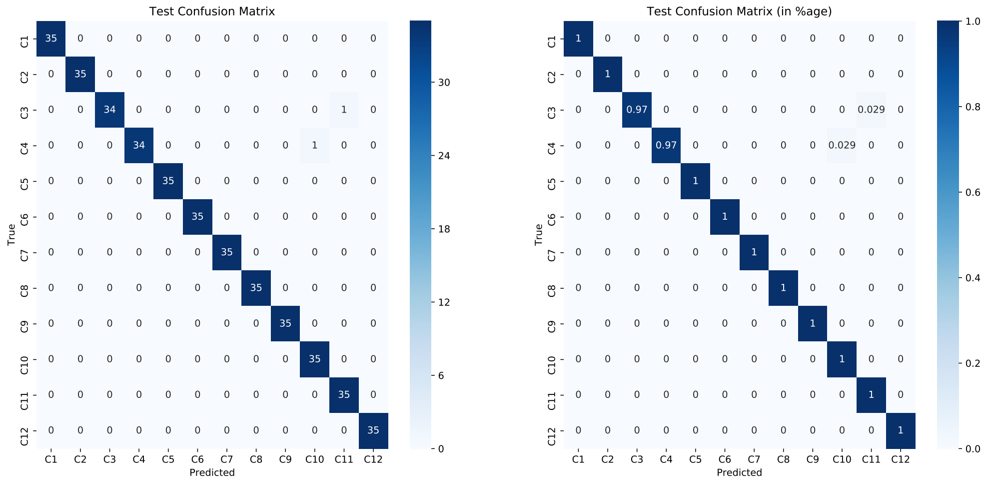

```
plt.figure(2,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x00000000265CB6A0>
```

```
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 0x0000000026E03B70>
```

```
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"
```

This accuracy is much better. It seems underlying assumption of LDA of using same covariance matrix for all classes is a valid one for this dataset.

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

```
data_without_normal = read.csv("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)),]
```

```
lda_fit_new = lda(fault~., train_data_new)
pred_train_new = predict(lda_fit_new, newdata = train_data_new)
pred_test_new = predict(lda_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)
```

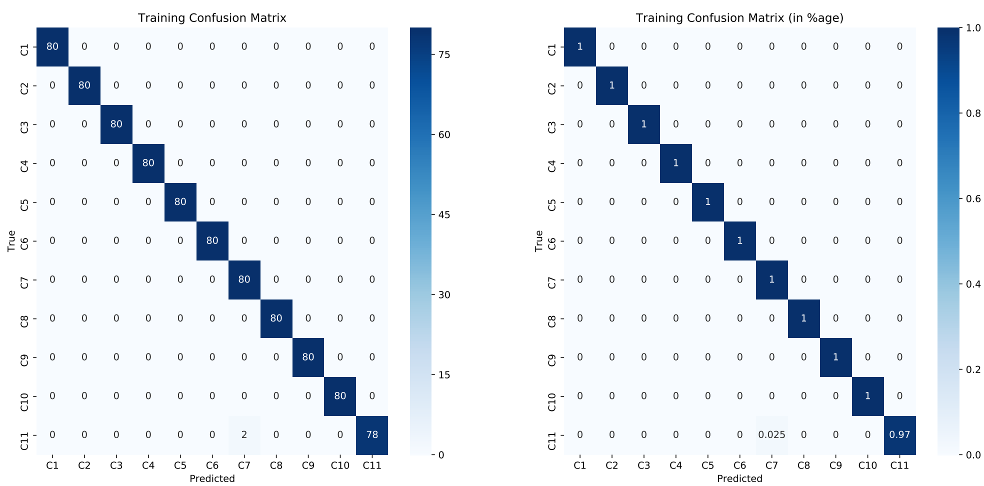
```
sns.heatmap(r.train_confu_new, annot= True,fmt = "d",
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000026E860F0>
```

```
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 0x0000000026E0CC18>
```

```
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")
```

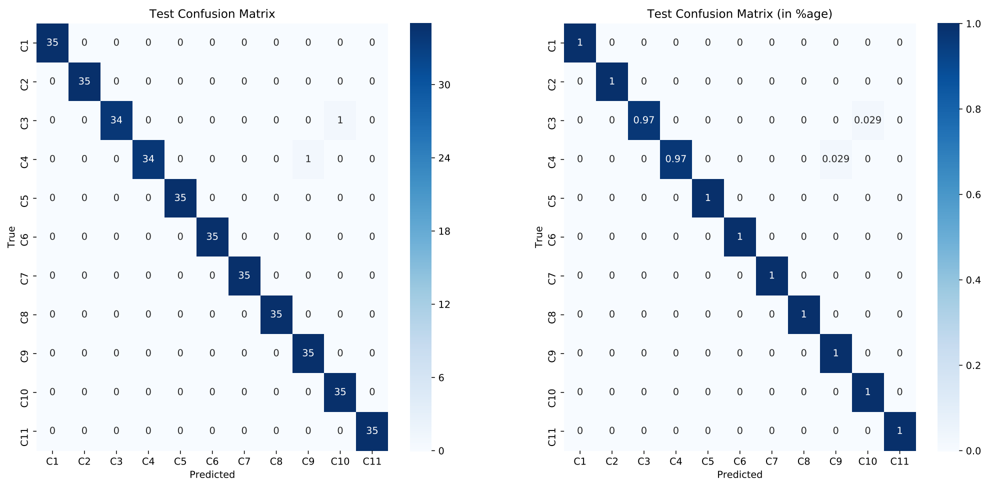
```
## <matplotlib.axes._subplots.AxesSubplot object at 0x00000000270B7860>
```

```
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 0x000000002729C3C8>
```

```
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.

Last updated: 7^{th} July, 2019