

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 from 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 10 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 : Inner race fault (0.007 inch)
- C5 : Inner race fault (0.014 inch)
- C6 : Inner race fault (0.021 inch)
- C7 : Normal
- C8 : Outer race fault (0.007 inch, data collected from 6 O'clock position)
- C9 : Outer race fault (0.014 inch, 6 O'clock)
- C10 : Outer race fault (0.021 inch, 6 O'clock)

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_48k_2048_load_1.csv',
                           header = T)
```

```

# Change the above line to include your folder that contains data
set.seed(1)
index = c(sample(1:230,75),sample(231:460,75), sample(461:690,75),
          sample(691:920,75),sample(921:1150,75),sample(1151:1380,75),
          sample(1381:1610,75),sample(1611:1840,75),sample(1841:2070,75),
          sample(2071:2300,75))

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.

```

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)

```

```

import seaborn as sns
import matplotlib.pyplot as plt
fault_type = ['C1','C2','C3','C4','C5','C6','C7','C8','C9','C10']
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 0x0000000018FD1448>
```

```

plt.title('Training Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.subplot(122)
sns.heatmap(r.train_confu/155, annot= True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")

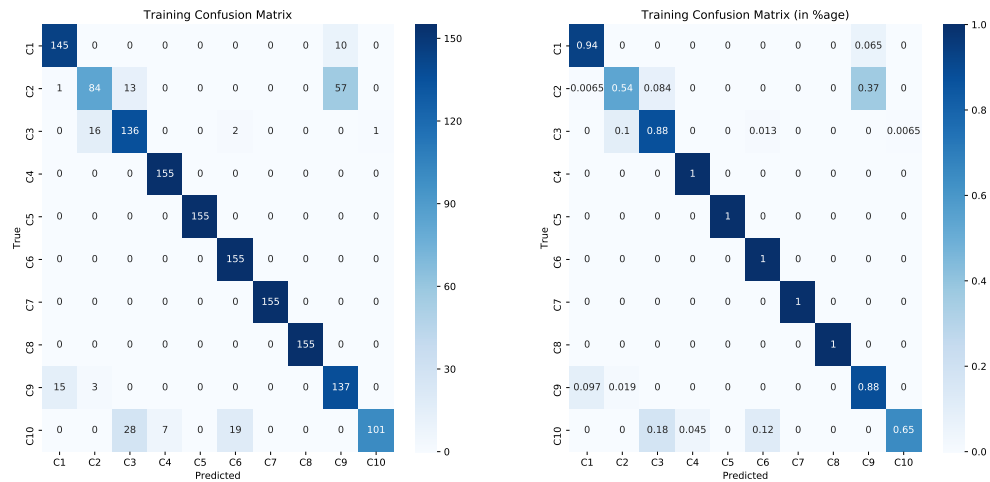
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000029BE4048>
```

```

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

```



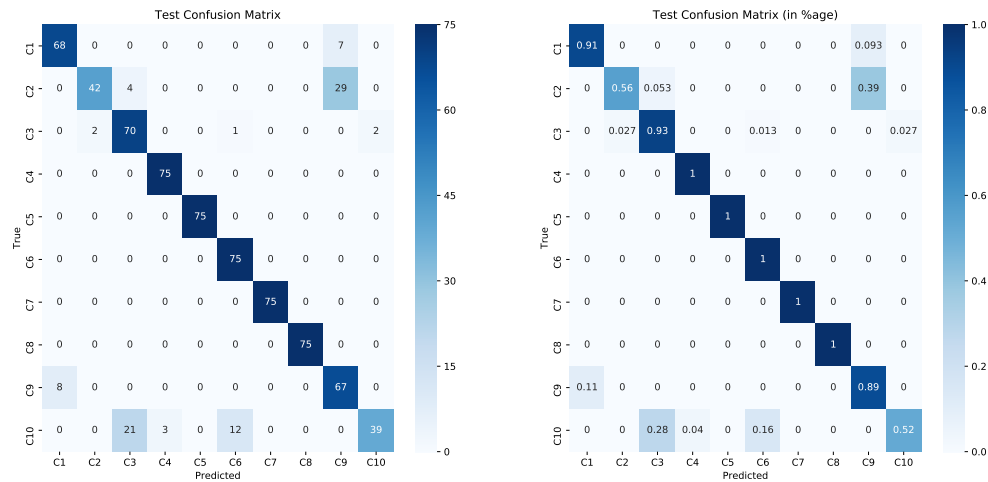
```
plt.figure(2,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")

## <matplotlib.axes._subplots.AxesSubplot object at 0x000000002A494188>

plt.title('Test Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('True')
plt.subplot(122)
sns.heatmap(r.test_confu/75, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")

## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000029C85388>

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



```
overall_test_accuracy = sum(diag(test_confu))/750
sprintf("Overall Test Accuracy: %.4f", overall_test_accuracy*100)
```

```
## [1] "Overall Test Accuracy: 88.1333"
```

We will see later that quadratic discriminant analysis gives much better results than this. So probably the assumption of using same covariance matrix for all classes is a bad one for this particular dataset.

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: 14th February, 2020