Bagging for Bearing Fault Classification

Biswajit Sahoo

Bagging (Bootstrap aggregating) is the process of combining results of several trees to come up with a classifier. Bootstrapping is used to generate several samples form the original training set. Trees are grown on each bootstrap sample. Finally the results of all trees are averaged to obtain final classifier. If several random variables are uncorrelated, averaging those reduces variance. Similarly if individual trees are not correlated to each other, bagging produces a result with low variance. In this post, we will apply bagging to classify different bearing faults.

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

```r
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.

```
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(342)
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. For bagging we will use ‘randomForsest’ package. As the name suggests, the package is also used for random forest classification. As we will see in another post, bagging and random forest are very similar to each other. For the time being, we will use that package for bagging.

```
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
bagging_fit = randomForest(fault~., train_data, mtry = 8)
pred_bagging = predict(bagging_fit, test_data)
# Confusion matrix
test_confu = table(test_data$fault, pred_bagging)
```

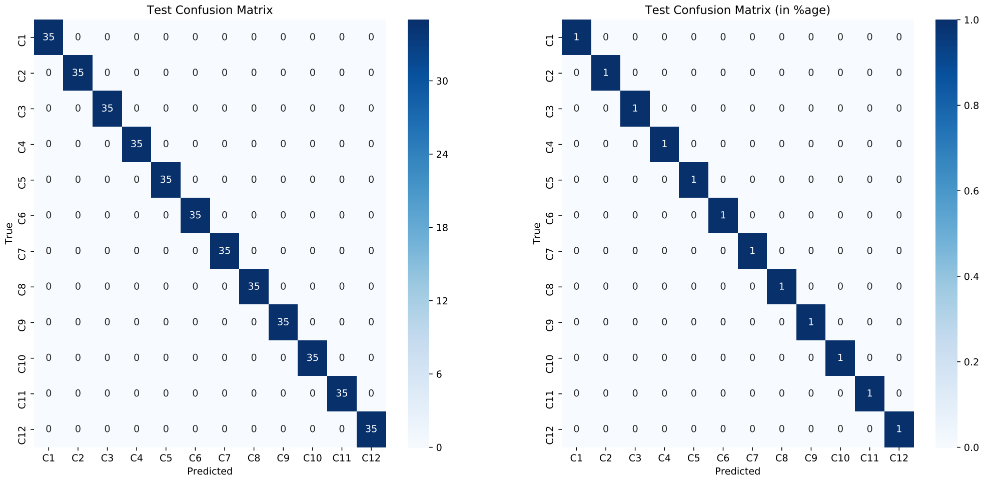
```
import seaborn as sns
import matplotlib.pyplot as plt
fault_type = [’C1’,’C2’,’C3’,’C4’,’C5’,’C6’,’C7’,’C8’,’C9’,’C10’,’C11’,’C12’]
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x000000001A736B88>
```

```
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 0x000000001A736848>
```

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



	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
C1	35	0	0	0	0	0	0	0	0	0	0	0
C2	0	35	0	0	0	0	0	0	0	0	0	0
C3	0	0	35	0	0	0	0	0	0	0	0	0
C4	0	0	0	35	0	0	0	0	0	0	0	0
C5	0	0	0	0	35	0	0	0	0	0	0	0
C6	0	0	0	0	0	35	0	0	0	0	0	0
C7	0	0	0	0	0	0	35	0	0	0	0	0
C8	0	0	0	0	0	0	0	35	0	0	0	0
C9	0	0	0	0	0	0	0	0	35	0	0	0
C10	0	0	0	0	0	0	0	0	0	35	0	0
C11	0	0	0	0	0	0	0	0	0	0	35	0
C12	0	0	0	0	0	0	0	0	0	0	0	35

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
C1	1	0	0	0	0	0	0	0	0	0	0	0
C2	0	1	0	0	0	0	0	0	0	0	0	0
C3	0	0	1	0	0	0	0	0	0	0	0	0
C4	0	0	0	1	0	0	0	0	0	0	0	0
C5	0	0	0	0	1	0	0	0	0	0	0	0
C6	0	0	0	0	0	1	0	0	0	0	0	0
C7	0	0	0	0	0	0	1	0	0	0	0	0
C8	0	0	0	0	0	0	0	1	0	0	0	0
C9	0	0	0	0	0	0	0	0	1	0	0	0
C10	0	0	0	0	0	0	0	0	0	1	0	0
C11	0	0	0	0	0	0	0	0	0	0	1	0
C12	0	0	0	0	0	0	0	0	0	0	0	1

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

```
## [1] "Overall Test Accuracy: 100.0000"
```

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(4)
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)),]
```

```
bagging_fit_new = randomForest(fault~., train_data_new, mtry = 8)
pred_bagging_new = predict(bagging_fit_new, test_data_new)
# Confusion matrix
test_confu_new = table(test_data_new$fault, pred_bagging_new)
```

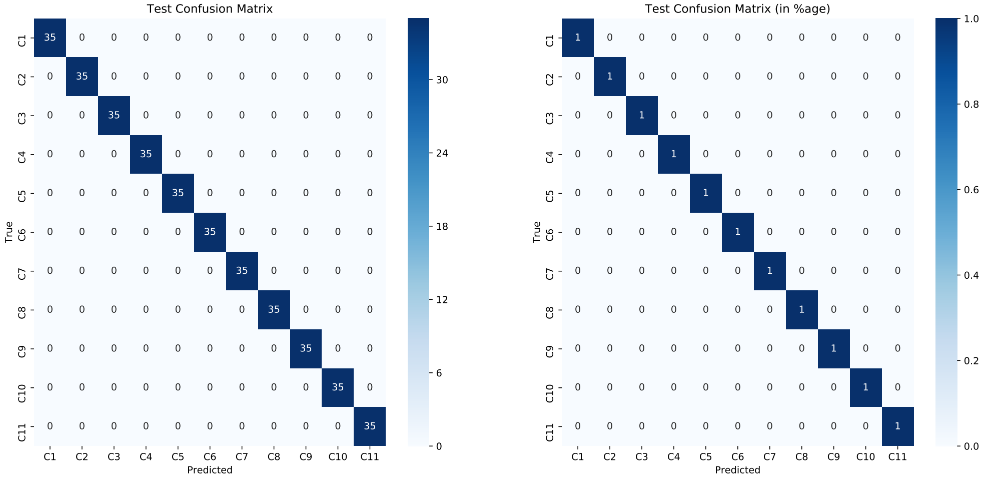
```
import seaborn as sns
import matplotlib.pyplot as plt
fault_type = ['C1','C2','C3','C4','C5','C6','C7','C8','C9','C10','C11']
plt.figure(1,figsize=(18,8))
plt.subplot(121)
sns.heatmap(r.test_confu_new, annot = True,
xticklabels=fault_type, yticklabels=fault_type, cmap = "Blues")
```

```
## <matplotlib.axes._subplots.AxesSubplot object at 0x0000000016F9A288>
```

```
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 0x000000002891EC88>
```

```
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: 100.0000"
```

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] randomForest_4.6-14 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: 15^{th} February, 2020