Accuracy_CI

Oliver Dürr

3/27/2019

Accuracy CI (not weighted)

Creation of some random data

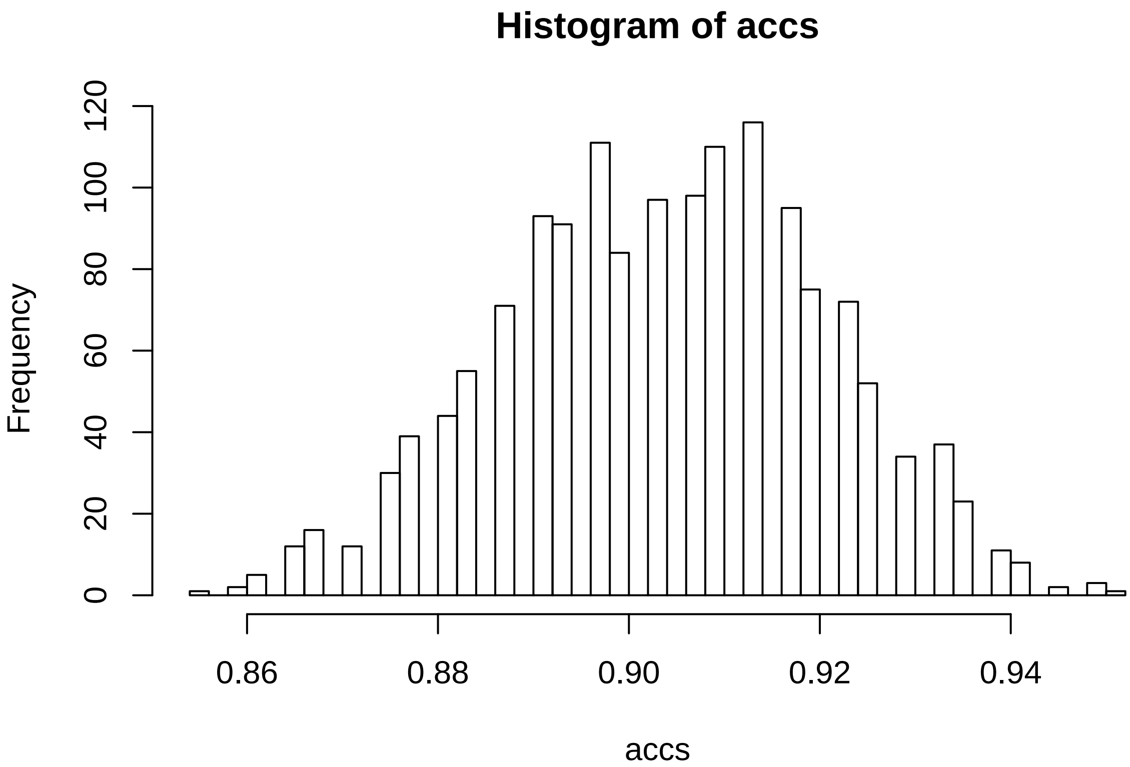
```
create_preds = function(y_true) {
  weights = nclasses/n
  y_pred = y_true
  for (i in 1:length(y_pred)) {
    # Get a random index in 20 precent
    if (sample(c(TRUE, FALSE),1, prob = c(0.2,0.8))) {
      y_pred[i] = sample(c(1,2,3), size = 1, prob=weights)
    }
  }
  return (y_pred)
}

nclasses = c(10,100,200) #Number of classes in the training set
n = sum(nclasses)
y_true = c(rep(1, nclasses[1]), rep(2, nclasses[2]), rep(3, nclasses[3])) #True_labels in the testset
y_pred = create_preds(y_true)
acc = mean(y_pred == y_true)
```

The single sample of random dataset has an accuracy of 0.8967742.

Getting the true distribution of the accuracy

```
accs = rep(NA, 1500)
for (r in 1:length(accs)) {
  y_pred = create_preds(y_true)
  accs[r] = mean(y_pred == y_true)
}
hist(accs, sqrt(length(accs)))
```




```
q2 = quantile(accs, 0.975)
q1 = quantile(accs, 0.025)
novel_sample= c(q1,q2)
ci_sample = q2 - q1
```

The sampled 95% CI should be approx from 0.8709677 to 0.9354839 and the range 0.0645161.

Calculating the CI for the accuracy (Wald Method)

First, we use the Wald intervall, see e.g. https://machinelearningmastery.com/report-classifier-performance-confidence-intervals/

```
acc_ci = acc + c(qnorm(0.025), qnorm(0.975)) * sqrt( (acc * (1 - acc)) / n)
cat(acc_ci, acc_ci[2]-acc_ci[1])
```

```
## 0.8629051 0.9306432 0.06773809
```

Calculating the CI using bootstrap

```
library(boot)
df = data.frame(y_true, y_pred)
acc.func = function(data, ind) {
  mean(y_true[ind] == y_pred[ind])
}
r.boot <- boot(df, R = 1000, statistic = acc.func)
r.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
```

```
##
## Call:
## boot(data = df, statistic = acc.func, R = 1000)
##
##
## Bootstrap Statistics :
##      original        bias    std. error
## t1* 0.9096774 -0.000816129  0.01613678
```

Calculation of the CI

We now calculate the CI from the bootstrap samples using different techniques:

- Assuming a normal distribution (like in the Wilson CI)
- Using the quantiles
- Using a method called BCA (I forget what that was)

```r
se = sd(r.boot$t)
ci_norm = r.boot$t0 + c(qnorm(0.025), qnorm(0.975)) * se
ci_quant = quantile(r.boot$t, c(0.025, 0.975))
d = boot.ci(boot.out = r.boot, conf = 0.95, type = "bca")

dd = boot.ci(boot.out = r.boot, conf = 0.95, type = "norm")
dd$normal[2:3] - ci_norm
```

```
## [1] 0.000816129 0.000816129
```

```r
ci_bca = c(d$bca[4],d$bca[5])
res_df = data.frame(ci_norm, ci_quant, ci_bca, wilson_ci = acc_ci, novel_sample)
res_df = rbind(res_df, res_df[2,] - res_df[1,])
rownames(res_df)[3] = 'range of CI'
res_df
```

```
##                ci_norm   ci_quant     ci_bca  wilson_ci novel_sample
## 2.5%         0.8780499 0.87419355 0.87096774 0.86290515   0.87096774
## 97.5%        0.9413049 0.93870968 0.93225806 0.93064324   0.93548387
## range of CI 0.0632550 0.06451613 0.06129032 0.06773809   0.06451613
```

Looks like that the bootstrap CI shows a lower accuracy than the Willson and Novel Sample

Accuracy CI (Weighted)

Now we calculate the weighted accuracy. We do this by calculating the accuracys for the individal classes.

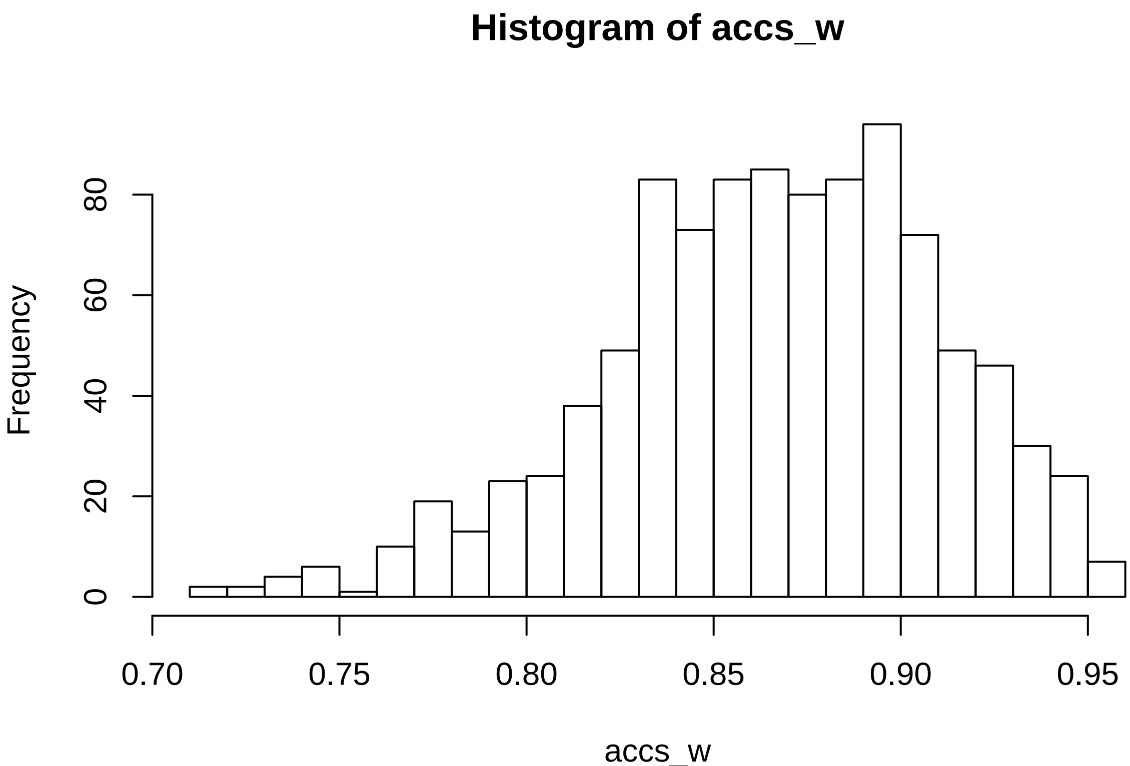
```r
# We use ind to compatible with the bootstrap version
acc.func = function(data, ind) {
  #ind = 1:310
  y_t = data$y_true[ind]
  y_p = data$y_pred[ind]
  acc = 0
  for (clazz in 1:3) {
    idx = y_t == clazz
    acc = acc + mean(y_t[idx] == y_p[idx])
  }
  return (acc / 3)
```

```
}
acc.func(df, 1:n)
```

```
## [1] 0.8466667
```

Sampling on new data

```
accs_w = rep(NA, 1000)
for (r in 1:length(accs_w)) {
  y_pred = create_preds(y_true)
  accs_w[r] = acc.func(data.frame(y_true, y_pred), 1:310)
}
hist(accs_w, sqrt(length(accs_w)))
```



And calulating the accuracy

```
q2_w = quantile(accs_w, 0.975)
q1_w = quantile(accs_w, 0.025)
novel_sample_w= c(q1_w,q2_w)
ci_sample = q2_w - q1_w
mean(accs_w)
```

```
## [1] 0.8663483
```

```
median(accs_w)
```

```
## [1] 0.87
```

The sampled 95% CI should be approx from 0.8709677 to 0.9354839 and the range 0.1717083.

Bootstrapping

```r
library(boot)
r.boot <- boot(data.frame(y_true, y_pred), R = 1000, statistic = acc.func)
r.boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = data.frame(y_true, y_pred), statistic = acc.func,
##     R = 1000)
##
##
## Bootstrap Statistics :
##      original       bias    std. error
## t1* 0.8266667 -0.001698419  0.05353752
```

```r
se = sd(r.boot$t)
ci_norm = r.boot$t0 + c(qnorm(0.025), qnorm(0.975)) * se
d = boot.ci(boot.out = r.boot, conf = 0.95, type = "bca")
ci_quant = quantile(r.boot$t, c(0.025, 0.975))
ci_bca = c(d$bca[4],d$bca[5])
res_df_w = data.frame(ci_norm, ci_quant, ci_bca,novel_sample_w)
res_df_w = rbind(res_df_w, res_df_w[2,] - res_df_w[1,])
rownames(res_df_w)[3] = 'range of CI'
res_df_w
```

```
##               ci_norm  ci_quant    ci_bca novel_sample_w
## 2.5%        0.7217350 0.7236019 0.7138228      0.7716250
## 97.5%       0.9315983 0.9276502 0.9255769      0.9433333
## range of CI 0.2098632 0.2040483 0.2117541      0.1717083
```