Improve and tidy estimation of optimal cutpoints

Christian Thiele & Gerrit Hirschfeld

06 July 2017
What do we want ‘optimal’ cutpoints for?

Binary classification via:

- Biological markers
- Psychological scores
- Model predictions

Independent variable
optimal cutpoint and distribution by class

![Graph showing count distribution by value for 'no' and 'yes' classes.](image-url)
Problems with ‘optimal’ cutpoints

- Prone to overfitting
- Selecting the ‘optimal’ cutpoint by trying out all possible ones leads to
  - overestimation of accuracy
  - highly variable cutpoints

95% confidence interval
of the 'optimal' cutpoint; empirically maximized metric
Some features of cutpointr

- More robust methods for lower variability of ‘optimal’ cutpoints
- Included bootstrapping (parallelizable)
- Extensibility by user-defined functions
- Tidy interface and output
Kernel method

- lower variability for maximizing sensitivity + specificity

Optimal cutpoint based on kernel smoothed densities

maximizing sensitivity + specificity
suicide %>%
  cutpointr(x = dsi, class = suicide,
           subgroup = gender,
           method = maximize_metric,
           metric = accuracy,
           direction = ">=",
           pos_class = "yes", neg_class = "no",
           boot_runs = 200)
Tidy interface and output

```r
suicide %>%
  cutpointr(x = dsi, class = suicide,
            subgroup = gender,
            method = maximize_metric,
            metric = accuracy,
            direction = ">=",
            pos_class = "yes", neg_class = "no",
            boot_runs = 200)
```
Tidy interface and output

The returned object is also a normal tibble

```r
> suicide %>% cutpointr(dsi, suicide, gender, boot_runs = 200)
Assuming yes as the positive class
Assuming the positive class has higher x values
# A tibble: 2 x 18

  subgroup direction optimal_cutpoint     method       Sun_Sens_Spec
   <chr>    <chr>            <dbl>        <chr>           
1 female   >=                2 maxmlze_metric 1.808118
2 male     >=                3 maximize_metric 1.625106

accuracy sensitivity specificity    AUC pos_class neg_class
   <dbl>       <dbl>       <dbl>     <dbl>       <fctr> <fctr>
1 0.8852041   0.9259259   0.8821918  0.9446474      yes  no
2 0.8428571   0.7777778   0.8473282  0.8617472      yes  no

prevalence outcome predictor grouping  data
   <dbl>    <chr>      <chr>           <chr>
1 0.06887755 suicide dsi gender <tibble [392 x 2]>
2 0.06428571 suicide dsi gender <tibble [140 x 2]>

roc_curve boot
          <list> <list>
1 <tibble [11 x 10]> <tibble [200 x 18]>
2 <tibble [11 x 10]> <tibble [200 x 18]>
```

Automatic 'guessing' of the positive / negative class and whether higher or lower predictor values imply the positive class

Data per group as nested tibbles

ROC curve and bootstrap results as nested tibbles
Tidy interface and output

The returned object is also a normal tibble

```r
> suicide %>% cutpoint(dsi, suicide, gender, boot_runs = 200)
Assuming yes as the positive class
Assuming the positive class has higher x values
# A tibble: 2 x 18
  subgroup direction optimal_cutpoint method     Sun_Sens_Spec
  <chr>   <chr>            <dbl>    <chr>         <chr>
1 female >=               2.00 maxlmlze_metric 1.808118
2 male >=                3.00 maximize_metric 1.625106
```

Data per group as nested tibbles

ROC curve and bootstrap results as nested tibbles
### Summary

```r
summary(cp)
```

<table>
<thead>
<tr>
<th>optimal_cutpoint</th>
<th>Sum_Sens_Spec</th>
<th>accuracy</th>
<th>sensitivity</th>
<th>specificity</th>
<th>AUC</th>
<th>n_pos</th>
<th>n_neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.7518</td>
<td>0.8647</td>
<td>0.8889</td>
<td>0.9238</td>
<td>36</td>
<td>496</td>
<td></td>
</tr>
</tbody>
</table>

**Observation**

<table>
<thead>
<tr>
<th>prediction</th>
<th>yes</th>
<th>no</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>32</td>
<td>68</td>
</tr>
<tr>
<td>no</td>
<td>4</td>
<td>428</td>
</tr>
</tbody>
</table>

**Predictor summary:**

<table>
<thead>
<tr>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max.</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.000000</td>
<td>0.000000</td>
<td>0.000000</td>
<td>0.9210526</td>
<td>1.0000000</td>
<td>11.0000000</td>
<td>1.8527143</td>
</tr>
</tbody>
</table>

**Predictor summary per class:**

<table>
<thead>
<tr>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>0</td>
<td>0</td>
<td>0.6330645</td>
<td>0</td>
<td>10</td>
<td>1.412225</td>
</tr>
<tr>
<td>yes</td>
<td>0</td>
<td>4</td>
<td>5.4888889</td>
<td>6</td>
<td>11</td>
<td>2.549821</td>
</tr>
</tbody>
</table>

**Bootstrap summary:**

<table>
<thead>
<tr>
<th>Min.</th>
<th>1st Qu.</th>
<th>Median</th>
<th>Mean</th>
<th>3rd Qu.</th>
<th>Max</th>
<th>SD</th>
</tr>
</thead>
<tbody>
<tr>
<td>optimal_cutpoint</td>
<td>1.0000</td>
<td>2.0000</td>
<td>2.0000</td>
<td>2.1950</td>
<td>2.0000</td>
<td>4.0000</td>
</tr>
<tr>
<td>Sum_Sens_Spec</td>
<td>1.3939</td>
<td>1.6442</td>
<td>1.7240</td>
<td>1.7072</td>
<td>1.7729</td>
<td>1.8778</td>
</tr>
<tr>
<td>Accuracy_b</td>
<td>0.7462</td>
<td>0.8534</td>
<td>0.8703</td>
<td>0.8623</td>
<td>0.8835</td>
<td>0.9267</td>
</tr>
<tr>
<td>Accuracy_oob</td>
<td>0.7143</td>
<td>0.8462</td>
<td>0.8639</td>
<td>0.8538</td>
<td>0.8758</td>
<td>0.9196</td>
</tr>
<tr>
<td>Sensitivity_b</td>
<td>0.7419</td>
<td>0.8680</td>
<td>0.8974</td>
<td>0.8985</td>
<td>0.9378</td>
<td>1.0000</td>
</tr>
<tr>
<td>Sensitivity_oob</td>
<td>0.5000</td>
<td>0.7857</td>
<td>0.8750</td>
<td>0.8531</td>
<td>0.9286</td>
<td>1.0000</td>
</tr>
<tr>
<td>Specificity_b</td>
<td>0.7321</td>
<td>0.8516</td>
<td>0.8663</td>
<td>0.8596</td>
<td>0.8824</td>
<td>0.9306</td>
</tr>
<tr>
<td>Specificity_oob</td>
<td>0.6979</td>
<td>0.8438</td>
<td>0.8620</td>
<td>0.8541</td>
<td>0.8780</td>
<td>0.9412</td>
</tr>
<tr>
<td>Kappa_b</td>
<td>0.2012</td>
<td>0.3679</td>
<td>0.4155</td>
<td>0.4127</td>
<td>0.4645</td>
<td>0.6042</td>
</tr>
<tr>
<td>Kappa_oob</td>
<td>0.1775</td>
<td>0.3413</td>
<td>0.3950</td>
<td>0.3878</td>
<td>0.4440</td>
<td>0.5455</td>
</tr>
</tbody>
</table>
User defined metric functions

The arguments to method and metric are actual functions

- metric is passed to method

accuracy

```r
## function(tp, fp, tn, fn, ...) {
##   Accuracy <- cbind((tp + tn) / (tp + fp + tn + fn))
##   colnames(Accuracy) <- "Accuracy"
##   return(Accuracy)
## }
## <environment: namespace:cutpointr>
```
Plots

cp <- cutpointr(suicide, dsi, suicide, gender, 
boot_runs = 200, 
direction = ">=", pos_class = "yes") 
plot(cp)
Single plots

```r
suicide %>%
cutpointtr(dsi, suicide, gender) %>%
plot_roc(display_cutpoint = FALSE)
```
Thank you

Not yet on CRAN but on Github:
https://github.com/Thie1e/cutpointr

Funding: BMBF Indimed