PAV & ROCCH

Antonio Bella
PAV & ROCCH

- Pool Adjacent Violators
  - *An Empirical Distribution Function for Sampling With Incomplete Information*
  - Miriam Ayer et al.

- Receiver Operating Characteristics Convex Hull
  - *Robust Classification for Imprecise Environments*
  - Foster J. Provost and Tom Fawcett
Reference Paper

- **PAV and the ROC Convex Hull**
- **Authors:**
  - Tom Fawcett
  - Alexandru Niculescu-Mizil
- **Publication:**
  - Machine Learning Journal
  - Volume 68, Number 1, July 2007
Motivation

- Classifier calibration
  - Convert scores into reliable probabilities
- Calibration technique
  - Isotonic regression (PAV)
- PAV algorithm is equivalent to ROCCH method.
Introduction 1

- The raw scores are not always good estimates of true probabilities.
- They must be **calibrated**.
- Technique based on isotonic regression:
  - Simple and effective.
  - It uses the **PAV algorithm** to find an isotonic (monotonically increasing) transformation of the classifiers scores that yields the lowest Brier Score (and cross-entropy).
Introduction II

- In other application domains, the scores are used for an ROC analysis.
- The initial purpose of the ROCCH was to facilitate the selection of the classifier that is optimal for given misclassification costs and the elimination of classifiers that are never optimal regardless of the misclassification costs.
Introduccion III

- PAV and ROCCH algorithms are equivalent.
- Generating the ROCCH of a single classifier can be viewed as finding an isotonic transformation of the scores that maximizes the area under the ROC curve (AUC).
Algorithm 1 Basic PAV method for generating probability estimates

**Input:** Scored training set \((f_i, y_i)\), where \(f_i\) is the score assigned by the classifier and \(y_i\) is the correct class.

**Output:** Stepwise constant function generated by \(m\)

1: begin
2: Sort training set instances increasing by \(f_i\)
3: Put each training instance in its own group, \(G_{i,i}\) and predict \(m_{i,i} = y_i\)
4: while \(\exists G_{k,i-1}\) and \(G_{i,l}\) ST \(m_{k,i-1} \geq m_{i,l}\) do
5: Pool the instances in \(G_{k,i-1}\) and \(G_{i,l}\) into one group, \(G_{k,l}\)
6: \(m_{k,l} = (\sum_{i=k}^{l} y_i)/(l - k + 1)\)
7: Predict \(m_{k,l}\) for all instances in \(G_{k,l}\)
8: end while
9: Output the stepwise constant function generated by \(m\)
10: end
# PAV example

<table>
<thead>
<tr>
<th>#</th>
<th>Score</th>
<th>Probabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Initial</td>
</tr>
<tr>
<td>0</td>
<td>0.9</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0.8</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0.7</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0.6</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0.55</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0.5</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>0.45</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>0.4</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>0.35</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0.27</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0.18</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0.1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>0.02</td>
<td>0</td>
</tr>
</tbody>
</table>
Algorithm 2 Generating an ROC curve

Inputs: $L$, the set of test examples; $f(i)$, the probabilistic classifier's estimate that example $i$ is positive, $P$ and $N$.
Outputs: $R$, a list of points defining the ROC curve.

1: begin
2: $L_{\text{sorted}} \leftarrow L$ sorted decreasing by $f$ scores
3: $fp\text{ rate} \leftarrow 0; tp\text{ rate} \leftarrow 0$
4: $R \leftarrow \emptyset$
5: $f_{\text{prev}} \leftarrow -\infty$
6: $i \leftarrow 1$
7: while $i \leq |L_{\text{sorted}}|$ do
8:  if $f(i) \neq f_{\text{prev}}$ then
9:   push ($fp\text{ rate}, tp\text{ rate}$) onto $R$
10:  $f_{\text{prev}} \leftarrow f(i)$
11: end if
12: if $L_{\text{sorted}}[i]$ is a positive example then
13:   $tp\text{ rate} \leftarrow tp\text{ rate} + 1/P$
14: else /*$i$ is a negative example*/
15:   $fp\text{ rate} \leftarrow fp\text{ rate} + 1/N$
16: end if
17: $i \leftarrow i + 1$
18: end while
19: push $(1, 1)$ onto $R$
20: end
Algorithm 3 ROC Convex Hull Algorithm for a single classifier

Input: Scored training set \((f_i, y_i)\), where \(f_i\) is the score assigned by the classifier and \(y_i\) is the correct class.

Output: Stepwise constant function generated by \(r\)

1: begin
2: Sort training set instances increasing by \(f_i\)
3: Put each training instance in its own group, \(G_{i,i}\)
4: Generate the ROC curve
5: while \(\exists G_{k,i-1}\) and \(G_{i,l}\) that generate a concavity in the ROC curve do
6: Pool the instances in \(G_{k,i-1}\) and \(G_{i,l}\) into one group, \(G_{k,l}\)
7: Let \(r_{k,l}\) ST \(f_k \leq r_{k,l} \leq f_l\)
8: Predict \(r_{k,l}\) for all instances in \(G_{k,l}\)
9: Regenerate the ROC curve
10: end while
11: Output the stepwise constant function generated by \(r\)
12: end
Equivalence between PAV and ROCCH

**Theorem:** For any binary classifier and for any training set \((fi, yi)\) the following are true:

1. The ROC graph of the PAV-transformed classifier is identical to the convex hull of the original classifier.
2. For each instance, the prediction made by the PAV-transformed classifier is equal to
   \[
   \frac{\text{slope} \cdot \text{skew}}{1 + \text{slope} \cdot \text{skew}}
   \]
   where slope is the slope of the segment on the convex hull of the original classifier the instance belongs to and skew is the ratio of positive to negative cases in the training set.
Conclusions

- Both algorithms generate the same groups of tied instances implying that the ROC graph of the PAV-transformed classifier is the convex hull of the original classifier.
- Extensions and insights for one technique can be applied to the other.
More, more, more…
## Score ties

<table>
<thead>
<tr>
<th>Score</th>
<th>Initial</th>
<th>Pre1</th>
<th>Pre2</th>
<th>a1</th>
<th>a2</th>
<th>b1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>2/3</td>
<td>2/3</td>
<td>2/3</td>
<td>3/4</td>
</tr>
<tr>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>2/3</td>
<td>2/3</td>
<td>2/3</td>
<td>3/4</td>
</tr>
<tr>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>2/3</td>
<td>2/3</td>
<td>2/3</td>
<td>3/4</td>
</tr>
<tr>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3/4</td>
</tr>
<tr>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>0.4</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>1/2</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Example with score ties

<table>
<thead>
<tr>
<th>Score</th>
<th>Initial</th>
<th>Pre1</th>
<th>Pre2</th>
<th>a1</th>
<th>a2</th>
<th>b1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0.8</td>
<td>0</td>
<td>0</td>
<td>2/3</td>
<td>2/3</td>
<td>2/3</td>
<td>3/4</td>
</tr>
<tr>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>2/3</td>
<td>2/3</td>
<td>2/3</td>
<td>3/4</td>
</tr>
<tr>
<td>0.8</td>
<td>1</td>
<td>1</td>
<td>2/3</td>
<td>2/3</td>
<td>2/3</td>
<td>3/4</td>
</tr>
<tr>
<td>0.6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3/4</td>
</tr>
<tr>
<td>0.5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>0.4</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>1/2</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>0.4</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>1/2</td>
<td>1/3</td>
<td>1/3</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**ROC Curve**

**Calibration Function**
Implementation in Java

```java
public static void calibrarPAV(
    double[] arrayV,
    int[] claseV,
    Vector<Double> vLimSup,
    Vector<Double> vLimInf,
    Vector<Double> vP
)
```

**INPUT:**
- Sorted array of scores (Validation)
- Array with binary class (1 YES, 0 NO)

**OUTPUT:**
- (Calibration function)
  - Probability associate between up and down score out bounds
Statistical Comparisons of Classifiers over Multiple Data Sets

Comparisons of Classifiers

2 Classifiers

Wilcoxon Signed-Ranks Test

Multiple Classifiers

Friedman Test

Control Classifier

Bonferroni-Dunn post-hoc Test

Nemenyi post-hoc Test

Function in R

```r
SCC <- function (fileName, numberClassifiers, controlClassifier=0, nameClassifiers=c(1:numberClassifiers), byRows=TRUE)
```

where "dataij" is the "data" for data set "i" and classifier "j"
Experiments (2 Classifiers) Calibration vs No Calibration

> source("C:\Documents and Settings\Administrador\Mis documentos\CEC\CompClass.R")
> SCC("DMCRMSingleWO2P.txt",2,0,c("SingleWONOCal2P","SingleWOCal2P"),FALSE)
- Wilcoxon Signed-Ranks Test:
  Both classifiers perform equally well with p=0.95
  Average Classifier SingleWONOCal2P :  187883.5
  Average Classifier SingleWOCal2P :  190676.5

> SCC("DMCRMSingleWI2P.txt",2,0,c("SingleWINOCal2P","SingleWICal2P"),FALSE)
- Wilcoxon Signed-Ranks Test:
  Both classifiers perform equally well with p=0.95
  Average Classifier SingleWINOCal2P :  191769.7
  Average Classifier SingleWICal2P :  191416.8

> SCC("DMCRMJointWO2P.txt",2,0,c("JointWONOCal2P","JointWOCal2P"),FALSE)
- Wilcoxon Signed-Ranks Test:
  The difference between the classifiers is significant with p=0.95
  Average Classifier JointWONOCal2P :  217788.3
  Average Classifier JointWOCal2P :  216696.3

> SCC("DMCRMJointWI2P.txt",2,0,c("JointWINOCal2P","JointWICal2P"),FALSE)
- Wilcoxon Signed-Ranks Test:
  Both classifiers perform equally well with p=0.95
  Average Classifier JointWINOCal2P :  206319.3
  Average Classifier JointWICal2P :  204457.3
Experiments (4 Classifiers)

> SCC("DMCRM100NoCal2P.txt",4,0,c("SingleW0","SingleWI","JointWO","JointWI"),FALSE)

Friedman Test (with the Iman and Davenport modification)
There are significant differences between some classifiers with $p=0.95$

- Nemenyi post-hoc test
  The difference between the classifiers SingleW0 and SingleWI is significant with $p=0.95$
  The difference between the classifiers SingleW0 and SingleWI is significant with $p=0.90$

The difference between the classifiers SingleW0 and JointWO is significant with $p=0.95$
The difference between the classifiers SingleW0 and JointWO is significant with $p=0.90$

The difference between the classifiers SingleW0 and JointWI is significant with $p=0.95$
The difference between the classifiers SingleW0 and JointWI is significant with $p=0.90$

The difference between the classifiers SingleWI and JointWO is significant with $p=0.95$
The difference between the classifiers SingleWI and JointWO is significant with $p=0.90$

The difference between the classifiers SingleWI and JointWI is significant with $p=0.95$
The difference between the classifiers SingleWI and JointWI is significant with $p=0.90$

Average Classifier SingleW0 : 187883.5
Average Classifier SingleWI : 191769.7
Average Classifier JointWO : 217788.3
Average Classifier JointWI : 206319.3
“That’s all Folks!”