Abstract

Class distribution skews in imbalanced datasets may lead to models with prediction bias towards majority classes. In this paper, we propose a simple and general-purpose evaluation framework for imbalanced data classification that is sensitive to arbitrary skews in class cardinalities and importances.

Key Design Principles

- **Simplicity:** It should be intuitive and easy to use and interpret.
- **Generality:** It should be general-purpose, i.e., (i) extensible to an arbitrary number of classes and (ii) customizable to any application domain.

Skew-Sensitive Evaluation Framework

**Weighted Balanced Accuracy (WBA)**

Suppose we are given a test dataset with $N$ data items and $C$ distinct classes: $N = \sum_{i=1}^{C} n_i$

Assume a classifier correctly predicts $p_i$ out of $n_i$:

$$\text{Accuracy} = \frac{\sum_{i=1}^{C} p_i}{N}$$

Macro-average of Accuracy:

$$\text{Balanced Accuracy} = \frac{1}{C} \sum_{i=1}^{C} \text{Accuracy}_i$$

Generalize into *Weighted Balanced Accuracy* by extending it with per-class importance weights $w_i$:

$$\text{Weighted Balanced Accuracy} = \sum_{i=1}^{C} w_i \times \text{Accuracy}_i$$

**Weight Customization**

Importance criteria = User-defined

Importance criteria = Rarity

Multiple importance criteria

$$w_i = \frac{1}{f_i \times \sum_{j=1}^{C} \frac{1}{f_j}}$$

Partially-defined importance criteria: support the case when not all of the class weights are supplied by the user.

**Model Training Improvement**

Our framework can be easily extended to other metrics such as Precision, Recall, and F-Score.

Using $Loss_i$ to denote the total loss incurred by all samples within class $i$, with our proposed class weights $w_i$, the model training loss:

$$\mathcal{L} = \sum_{i=1}^{C} w_i \times Loss_i$$

Use Case 1: Learned Log Parsing

(a) macOS (skew = 8.454)

(b) BGL (skew = 8.900)

(c) Android (skew = 4.822)

(d) HDFS (skew = 0.202)

Use Case 3: URL Classification

<table>
<thead>
<tr>
<th>Category</th>
<th>#URLs</th>
<th>Rarity $\omega_i$</th>
<th>User $\omega_i$</th>
<th>Classification Accuracy</th>
<th>Ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>benign</td>
<td>16762</td>
<td>0.04</td>
<td>0.05</td>
<td>0.761</td>
<td>DBAC</td>
</tr>
<tr>
<td>NSFW</td>
<td>5276</td>
<td>0.14</td>
<td>0.05</td>
<td>0.965</td>
<td>DBAC</td>
</tr>
<tr>
<td>malware</td>
<td>1913</td>
<td>0.38</td>
<td>0.38</td>
<td>0.900</td>
<td>DBAC</td>
</tr>
<tr>
<td>phishing</td>
<td>1675</td>
<td>0.44</td>
<td>0.44</td>
<td>0.968</td>
<td>DBAC</td>
</tr>
</tbody>
</table>

Above: Evaluating and ranking the URL classification services

Right: Training and evaluating a URLNet model using WBA

Use Case 2: Sentiment Analysis

Table: Amazon per-class breakdown

<table>
<thead>
<tr>
<th>Class</th>
<th>Frequency</th>
<th>Weights</th>
<th>LSTM</th>
<th>RNN</th>
<th>GRU</th>
<th>BILSTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.092</td>
<td>0.7</td>
<td>0.19</td>
<td>0.04</td>
<td>0.16</td>
<td>0.17</td>
</tr>
<tr>
<td>2</td>
<td>0.052</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>3</td>
<td>0.075</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>4</td>
<td>0.142</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>5</td>
<td>0.639</td>
<td>0.3</td>
<td>0.81</td>
<td>0.96</td>
<td>0.84</td>
<td>0.83</td>
</tr>
</tbody>
</table>

(a) WBA vs. Other Metrics (Train+Test)

(b) WBA: Test vs. Train+Test